AD-A103 398 AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH SCHOO--ETC F/G 9/2
AN ANALOG SPEECH I/O CHANNEL FOR THE NOVA 2 COMPUTER BASED ON T--ETC(U)
MAR 81 D FREDAL, G C BEASLEY
UNCLASSIFIED AFIT/GE/EE/81M-2 NL

1 OF 2
AD A
103398

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY (ATC)

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

(1) LEVEL II

AN ANALOG SPEECH I/O CHANNEL FOR THE NOVA 2 COMPUTER BASED ON THE S-100 BUS

THESIS

AFIT/GE/EE/81M-2

Dan Fredal
Capt USAF

George C. Beasley, Jr.
Capt USAF

Approved for public release; distribution unlimited.

AFIT/GE/EE/81M-2

AN ANALOG SPEECH I/O CHANNEL FOR THE NOVA 2 COMPUTER BASED ON THE S-100 BUS

THESIS

Presented to the Faculty of the School of Engineering
of the Air Force Institute of Technology
Air University
in Partial Fulfillment of the
Requirement for the Degree of
Master of Science in Electrical Engineering

by

Dan Fredal, B.S.E.
Capt USAF

and

George C. Beasley, Jr., B.S.E.E.
Capt USAF
Graduate Electrical Engineering

March 1981

Preface

This thesis contained generous amounts of hardware and software engineering for two quite different computer systems. Accomplishing a project of this magnitude as a team effort is quite probably the only timely way to achieve a satisfactory capability. Since we both enjoy working with projects that involve hardware and software engineering, we feel that this thesis was well suited for us and that our team effort has provided the best possible outcome. Nevertheless, we feel the project was quite challenging and very interesting.

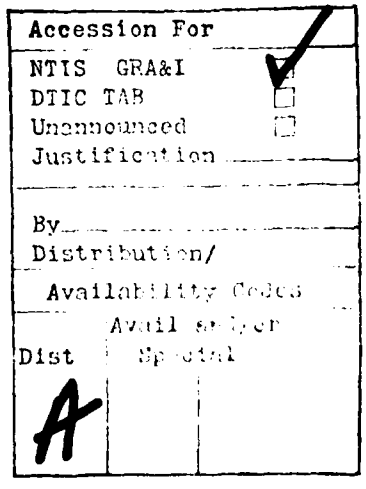
To interface the 8-bit Cromemco S-100 bus system to the 16-bit Nova system required developing considerable insight into the detailed operation of the hardware and software of both systems. As usual in projects of this type the hardware development went rather quickly. Most of our time was spent developing the Nova and Cromemco software.

To provide a systematic approach to solving possible problems, Captain Beasley concentrated most of his effort toward the Cromemco portion of the project while Captain Fredal devoted his effort toward the Nova. We both spent a large part of our time working together during the development of the Channel Protocol. With the concentrated knowledge each of us had of our system, we were able to resolve potential conflicts in the protocol that could have caused degraded operation within either of the systems.

We would like to express our thanks to our advisor, Dr. Matthew Kabrisky of the Air Force Institute of Technology, who fueled this thesis with his over-whelming inspiration. Deep appreciation is expressed for the help that we received from Captain Larry Kizer and his crystal ball. His uncanny talent for anticipating future hardware needs, coupled with his knowledge of the Nova 2 system, made the timely completion of this thesis possible.

Also, it is extremely important that we express our sincere gratitude to Serenna Beasley for the many diligent hours she spent proofreading and typing this manuscript.

George Beasley
Dan Fredal



Contents

List of Figures

Abstract

As a result of the software and hardware developed under this thesis, the AFIT Speech Lab's Cromemco Z-2 microcomputer system now serves as an I/O Channel for the Lab's Nova 2 minicomputer system. Since the Cromemco utilizes the S-100 bus, the I/O Channel provides considerable interface flexiblity for the Nova from the large number of relatively inexpensive interfaces available for S-100 based systems.

The hardware developed to connect the two systems allows 16-bit information to be transferred between both systems even though the Cromemco is based on an 8-bit microprocessor. In addition to this hardware an I/O Channel Protocol was developed that allows the Nova to initiate tasks in the Cromemco and to command data transfers between itself and the Cromemco system. The necessary software was also developed for both systems to implement this protocol. The Nova's software is structured so that new capabilities can be added in the Cromemco portion of the I/O channel without the necessity of making changes to the Nova software. The software for the Cromemco is a complete Channel Operating System that allows task modules to be added as separately linkable routines when new requirements are placed upon the I/O Channel. These modules can be easily added and should not require changes to the primary operating system that handles the basic I/O Channel Protocol.

AN ANALOG SPEECH I/O CHANNEL FOR THE NOVA 2 COMPUTER BASED ON THE S-100 BUS

I. Introduction

This thesis involves the design of an interface between the Cromemco Z-2 microcomputer and a Nova 2 minicomputer. When this interface is used with the software written for both systems, it provides an I/O Channel for the Nova. This was done initially to provide a speech I/O capability for the system. The requirement was later upgraded to provide a generalized I/O Channel capability. Chapter One discusses the background for the project and the basic configuration of both systems. Chapter Two provides insight into the initial problems that were addressed and the final design that was implemented. Chapter Three includes the actual design implementation and the theory of operation of the I/O Channel. Chapter Four discusses the conclusions and recommendations of the design team.

Background

The Electrical Engineering department of the Air Force Institute of Technology (AFIT) wishes to develop a general purpose, digital speech processing facility. This facility already includes an Eclipse S/250 minicomputer, a Nova 2 minicomputer with an expansion chassis and a Cromemco Z-2 microcomputer system. Both minicomputers are 16-bit machines built by the Data General (DG) Corporation. The Cromemco system has a Z-80 Central Processing Unit (CPU) and

is based on the S-100 bus. The two minicomputers share a ten megabyte hard disk (Model No. 6045) through an Inter-Processor Buffer (IPB) which arbitrates simultaneous accesses. The system also includes a tape drive (Model No. 6021), two terminals, and a high speed dot matrix printer.

One of the first tasks that the facility is required to support is the development of a speech encoding algorithm which will reduce the number of bits required to transmit digitized speech. This will allow increased rates of speech transmission over currently available communications networks. Another task will be to develop highly accurate speech recognition algorithms for an aircraft cockpit capable of responding to a pilot's verbal commands. In the past all digital speech work at AFIT was done on a remote mainframe computer. Unfortunately this technique involved intolerable turnaround delays and resulted in the development of a dedicated speech facility. With a dedicated processing capability, these delays could be greatly reduced to allow nearly real-time processing to be accomplished.

It was decided to use a Cromemco Z-2 system to provide the speech data acquisition capability for the facility, even though DG manufacturers such peripherals. There were three reasons for this decision. First, the DG peripherals required more funds than had been allocated. Obtaining additional funding would have resulted in a delay that would have caused most thesis work in the lab to stop until the

1981 cycle. Second, hardware for the S-100 bus, capable of performing a wide range of functions, is available from many manufacturers. This reduces the cost of the system itself and the cost of adding enhanced capabilities in the future. Third, the S-100 bus in the Cromemco makes future upgrade in capability a relatively easy task. This is due to the large number of boards currently manufactured for the bus and the fact that most upgrades simply require the purchase of the hardware and development of software drivers for it.

The Cromemco system includes a Cromemco Z-80 CPU Board and 4FDC Disk Controller Board; a Seimens 5-inch mini-floppy Disk Drive; two Seattle Computing 16/8 RAM 16K memory boards and a CPU Support Card; and a Techmar S-100 A/D Board, D/A Board, and Video Digitizer. The Video Digitizer was used by another thesis project. The 16/8 RAM memory meets the IEEE standard for the S-100 bus and allows both 16- and 8-bit data transfers. The CPU Support Card provides crystal controlled timers, interrupt controllers, a parallel input and output (I/O) port, and a bidirectional serial port. The S-100 A/D board uses an Analogic MP6812 16-channel A/D converter with a resolution of 12 bits and an approximate conversion time of 35 microseconds. The D/A board has four separate D/A converters (DAC-80-CBI-V). Each of these D/A converters has a resolution of 12 bits and a conversion time of approximately 3 microseconds.

Problem

The problem solved by this thesis was the design and implementation of an S-100 based I/O Channel with an initial capability of speech I/O for the Nova 2. The key word here is "Channel." Under this concept the Nova is required to initiate all I/O transactions. Also, the Cromemco must be completely transparent to the Nova user when it is involved in an I/O process. The only concern the user should have is to insure the Cromemco's power is turned on.

An additional requirement for this project was to insure that future upgrades to the system are as simple as possible. Such upgrades could be very complex processes if knowledge of both the Nova and Cromemco assembly languages and operating systems were required. Therefore, the interface and associated software were to be designed to allow upgrades with no software changes required in the Nova. This can allow a person who is only familiar with the Z-80 processor to accomplish the system modification or enhancement. In addition, the operating system developed for the Cromemco must be easy to enhance. It should not be necessary to change the operating system in order to add a new capability to the I/O Channel.

General Approach

First, the hardware interfaces to connect the two systems were designed and built. These interfaces had to provide for both programmed I/O and Direct Memory Access

Problem

The problem solved by this thesis was the design and implementation of an S-100 based I/O Channel with an initial capability of speech I/O for the Nova 2. The key word here is "Channel." Under this concept the Nova is required to initiate all I/O transactions. Also, the Cromemco must be completely transparent to the Nova user when it is involved in an I/O process. The only concern the user should have is to insure the Cromemco's power is turned on.

An additional requirement for this project was to insure that future upgrades to the system are as simple as possible. Such upgrades could be very complex processes if knowledge of both the Nova and Cromemco assembly languages and operating systems were required. Therefore, the interface and associated software were to be designed to allow upgrades with no software changes required in the Nova. This can allow a person who is only familiar with the Z-80 processor to accomplish the system modification or enhancement. In addition, the operating system developed for the Cromemco must be easy to enhance. It should not be necessary to change the operating system in order to add a new capability to the I/O Channel.

General Approach

First, the hardware interfaces to connect the two systems were designed and built. These interfaces had to provide for both programmed I/O and Direct Memory Access

(DMA) of data from the Nova. Status and command words were to be passed between the two systems by programmed I/O with bulk data transfers being done with DMA. The DMA was to be between the Nova and the latches in its interface and **not** between the Nova and Cromemco memories. A DMA directly into the Cromemco memory was not necessary to meet the requirements for speech I/O. It should be noted that DMA within the Nova is referred to as a Data CHannel (DCH) by DG. DCH will therefore be used to signify the DMA operation of the Nova to distingush it from a DMA within the Cromemco system.

Next a primitive means of speech I/O was developed to allow other students working on speech processing oriented theses to obtain the data they required. This preliminary speech I/O was accomplished by using programmed I/O and by buffering the data in the Cromemco rather than using the DCH. The DCH would have taken much too long to develop since it required both hardware and software development for the Cromemco to employ this capability. The major disadvantage of this first implementation was that buffering the speech data in the Cromemco limited the amount of data which could be converted at any one time to three seconds. Also, without the operating system in the Cromemco, a separate program had to be run in each computer to convert the speech.

Finally, the operating system was written for the Cromemco and software was written for the Nova, which

includes implementation of the DCH transfer of speech data. This allowed the speech data to be passed to and from the Nova's disk under full control of the developed software. This final implementation accomplished the requirements that were established for the I/O channel.

II. Detailed Analysis

This chapter briefly describes the Nova and Cromemco systems. It includes a discussion of the interface hardware and software designs necessary for both systems to be interconnected. Also included is a description of the speech channel communication protocol that was required.

Nova System

The Nova minicomputer consists of a chassis with a built-in power supply and a backplane with ten slots into which a CPU board, memory boards, and I/O interface boards can be inserted. Each slot has two 100-pin edge-connect sockets to connect the boards to busses and control lines which are on the backplane. Forty-eight of the 100 pins are reserved to carry signals to paddle boards located along the back edge of the backplane. Peripheral devices are connected through these paddle boards to the circuitry of the I/O interface boards. Since all ten slots in the Speech Lab's Nova were already in use, an expansion chassis was added. It provides ten additional slots which can be used by I/O interface boards and includes its own power supply. The I/O bus in the main chassis is connected to the expansion chassis's backplane through the appropriate connector and a set of buffers. There are two paddle boards built into the backplane of the expansion chassis, one for slot one and one for slot ten. The I/O boards perform equally well in the expansion chassis as in the main chassis except

for those requiring the high speed Data CHannel (DCH). Only the low speed DCH can be implemented from the expansion chassis.

DG has succeeded in creating an I/O system that is relatively easy to interface to as well as to program. In order to better understand how the Nova I/O system works, it is necessary to examine the structure of the Nova's I/O bus. It is a well designed, relatively uncomplicated system consisting of a 16-bit bidirectional data bus, a 6-bit device select bus, seven DCH control lines and fifteen control signals to handle programmed I/O and interrupts. DG refers to an I/O interface board and its associated peripheral as a device. Therefore, all future references to a device assume this convention. The processor selects a device by placing its code on the device select bus. The device decodes the device select bus and generates a device select signal which is used to gate all control signals into or out of the I/O interface board. Each device can have up to three separate ports (A, B, and C) with which to communicate with the system. Each port is generally composed of two separate 16-bit registers, one for output and one for input.

Each device has a BUSY flag and a DONE flag associated with it. The BUSY flag is set when the device is "started" or put to work on a task by the processor. When the device has completed the task it sets the DONE flag. The processor

can examine each flag individually. It can set the BUSY flag, which also clears the DONE flag, or it can clear both flags simultaneously. When both flags are cleared the device is considered idle.

The data transfer capabilities of the I/O bus includes both programmed I/O and the DCH. Programmed I/O is handled directly by the Nova processor, while the DCH is handled by hardware designed to maximize the speed of the transfer. Normally, programmed I/O is used to transfer data when it is necessary for the processor to examine or process each word as it is passed through the interface or when it is necessary to handshake with the device after each data transaction. The BUSY and DONE flags are used to synchronize the transfer of data by programmed I/O.

Since the DCH is a much faster data transfer operation than programmed I/O, it allows a large amount of data to be transferred in a very short period of time. There are two speeds under which DCH transfers can be made. The high speed DCH transfers data at a maximum rate of approximately 1,250,000 words per second, while the maximum rate for low speed DCH transfers is approximately 475,000 words per second. These speeds are achieved through the direct transfer of data between the Nova's memory and a device's port. The initiation and direction, either input or output, of a DCH is set by the device. However, the parameters of the transfer must first be set up by the Nova processor.

The Nova must load the address register of the device with the starting address of the area of memory to be transferred. It can also load the two's complement of the number of words to be transferred into the word count register of the device. Both registers are incremented by one each time a word is transferred. The processor uses the address register of the interface to determine the memory location involved in the transfer of each word. Since the device initiates the DCH request for each transfer, it can use the word count register to determine when the transfer is complete. Priority for the Data Channel is established by daisy chaining the Data Channel Priority control line. The device which is physically closest to the processor in the chain receives the highest priority.

Although the Nova I/O bus can handle interrupts, this capability was not required in the implementation of the I/O channel; therefore, Nova interrupts will not be discussed.

Cromemco System

Since both the Nova and Cromemco systems are stand alone computers, they appear similar in many respects, but they are actually quite different. The Cromemco system consists of a chassis housing a built-in power supply and an S-100 bus mother-board made up of twenty-one 100-pin connectors. Each of these connectors provides a slot into which S-100 compatible circuit cards can be inserted. Unlike the

Nova, each of the slots on the Cromemco mother-board receives the same set of signals; therefore, any circuit card that is compatible with the S-100 bus can be plugged into any of the slots. Consequently, there are no special or reserved slots in the Cromemco system. The only consideration that must be made is that no more than one card can be assigned to the same memory and/or I/O addresses. The built-in power supply provides three unregulated voltages (+8, -18, and +18 volts) to the bus. It is the responsibility of each circuit card inserted into a slot to have its own on-board regulators capable of supplying the power required by the circuit.

The S-100 bus has become one of the most popular hobby busses and is now considered by most as an industry standard. It was first introduced by MITS, Inc. (now part of the Pertec Computer Co.) on their 8080-based Altair computer. Since it was originally designed to support the 8080 processor, most of the bus signals are representative of the signals generated by an 8080 processor. Almost all of the 100 lines of the bus have a standardized predefined function (see Appendix A). Basically, the lines can be divided into four major groups. These groups consist of power and ground lines, address lines, data lines, and control lines.

The power and ground line group provides the voltages required by the computer. Six lines are assigned to provide

the three unregulated voltages and their ground returns. The address line group provides the sixteen lines required to supply a 16-bit memory address. The data line group is composed of sixteen lines and is used to supply program instructions and data. Normally, the data group lines are separated into two sub-groups of eight lines each. One set is used to supply data to the CPU, while the other set is used for data eminating from the CPU. A recent document that the Institute of Electrical and Electronics Engineers (IEEE) has published allows these sixteen lines to become bidirectional when the proper control signals are provided (see Appendix A). This allows the bus to be used with the evolving 16-bit micro-processors. This document is heralded as a standard and should serve to standardize the bus across the industry for both 8-bit and 16-bit operation. The remaining sixty-two lines comprise the control line group. These lines are used to carry timing and control signals between the CPU, memory and I/O.

With the already overwhelming acceptance of the S-100 bus, made evident by the extent of its use, it is an ideal choice as the computer bus for the I/O channel. There are numerous manufacturers building cards that comply with the standard, making the task of finding a commercially built card, to do all but the most specialized functions, quite easy. Since many manufacturers are generating large quantities of cards, the cost is considerably less than an equivalent device designed specifically for the Nova bus.

Since the S-100 bus is a true parallel bus, all signals within the system are available to any card placed on the bus. This provides for nearly unlimited use of these signals by an interface or I/O peripheral. The Nova's bus tends to be more restrictive in this case, since it requires certain modules to be placed into certain slots. Since the S-100 bus is not as tightly structured as the Nova's bus, the Cromemco system handles I/O differently than the Nova. The Cromemco's S-100 bus is capable of communicating directly with 256 I/O ports using the low byte of its address bus rather than a special device select bus like the Nova. The S-100 bus has two control lines that delineate I/O operations from normal memory transactions. One of these lines, sINP, indicates an input operation while the other, sOUT, indicates an output operation. Each interface must decode the least significant byte of the 16-bit address bus to determine its port addresses when either of these two I/O signals appear. Two other control signals, pWR and pDBIN, are always generated during memory write and read operations, respectively. These signals are used to tell the interface circuit when it must either supply data to or take data from the system.

The CPU Support Card was selected from the commercially available interfaces for the S-100 bus to provide the Cromemco with a set of crystal controlled timers and interrupt control circuitry. The timers which are accurate

to 250 nanoseconds and the interrupt circuitry are used to maintain the precise timing required for sampling analog speech or reconstructing digitized speech. Since slight variations in the sampling rate can cause the speech to become distorted, it is imperative that the sampling rate is kept constant when accomplishing either of these operations. To insure an exact rate, a crystal controlled timer on the CPU Support Card is used to interrupt the system when each sample must be taken. These interrupts cause the CPU to service the appropriate A/D or D/A converter at programmable time intervals set for the required sampling rate.

Nova Interface Hardware

The interface circuit for the Nova was built on a DG General Purpose Interface (GPI) board. This board provides circuitry which can be used to implement most of the three previously described ports, the BUSY and DONE flags, the interrupt control function and the DCH control function. In addition, the board has a breadboard area to allow the addition of custom circuitry. The pre-wired circuits, along with any required custom circuitry, must be connected together to create the desired interface for the peripheral device.

In order for the GPI to serve as an interface to the Cromemco, the on-board registers had to be assigned to the device's ports. The assignments were made by connecting the

control signal of a particular port to the clocks of each input and output register. The DCH address register is pre-wired on the board and was assigned to the B Port. This register is different from the others because it is a single register serving as both an input and output register for the Nova. This allows the Nova to load the B Port before the first DCH is attempted and to use its contents as the memory address for the data to be transferred during each DCH. The device, on the other hand, can only read this register, not load it. This register can also be incremented, which is a capability that a DCH address register must possess. An incrementable input register is also available on the GPI and was assigned to the C Port. This register was intended to be used as the word counter during a DCH but the Cromemco tracks the count internally, so it is used simply as an input register. The remaining pre-wired input register and output register were assigned to the A Port. At this point, only the A Port provides a complete bidirectional path to the Nova processor. In order for programmed command I/O word to be transferred to the Cromemco while DCH is in operation, two completely separate bidirectional paths to the device were required. Since the A Port is fully utilized during a DCH, an output register was added to the GPI for the C Port in order to make it bidirectional. The additional circuitry to accomplish this was added using the breadboard area provided on the GPI.

A paddle board on the backplane of the Nova expansion chassis is used to connect the Nova to the Cromemco. Since there are only forty-eight connections available on a paddle board and some of these are needed for control purposes, no more than two sets of sixteen lines can be used for the data transfer path. With this constraint, a method of transferring the three 16-bit ports over two 16-bit paths had to be devised. Each path could have been made bidirectional, which would have provided two completely separate data paths, or two unidirectional paths could have been created by allocating one path for input and the other path for output. The two unidirectional path approach was chosen since this appeared to simplify the circuitry in both the Nova and Cromemco interface boards. With this configuration there are no conflicts for the data path; therefore, no additional data path arbitration circuitry is required. All three ports are made available to the Cromemco by tying all of their inputs to the input data path and multiplexing all of their outputs to the output data path. This means that the B Port is readily available should it be required by the Cromemco at a later date. As indicated earlier, this port cannot be loaded by the device, but the GPI breadboard area has ample space for adding this capability at a later date. Cromemco data is demultiplexed from the input data path with the DATA CLOCK signal. This signal clocks the data into the input register selected by the Cromemco via one of the control lines on the interface. The output registers are

multiplexed onto the output data path through three sets of tristate bus drivers. The Cromemco selects the desired Nova output register by enabling its set of tristate drivers with one of the interface control lines. The register multiplexing/demultiplexing circuitry was put in the breadboard space provided on the GPI board.

The GPI device select circuitry was set up to decode 25 (octal) as a device select; therefore, the Cromemco appears to the Nova as device number 25. This number was selected to eliminate possible conflicts with existing devices. The BUSY and DONE flags were utilized to coordinate the transfer of data between the two computers. The hardware to provide the flags and allow them to function as prescribed by DG was already available on the GPI. These flags were also sent to the Cromemco so they can be monitored. In addition, the DEVICE COMPLETE input to the Nova was made available to the Cromemco. It is on the low to high transition of this signal that the DONE flag is set and the BUSY flag is cleared.

Cromemco Interface Hardware

The Cromemco interface card provides the necessary I/O ports to allow the 8-bit Cromemco Z-80 CPU to read or write to the 16-bit registers on the Nova interface board. It also provides an additional input port to allow monitoring of the Nova's BUSY and DONE flags and two additional output ports. One of these output ports allows the Cromemco to

initiate a DCH and the other contains an 8-bit latch with one of its bits used to inhibit the A Port DEVICE COMPLETE signal during DCH data transfers and two other bits used for selecting the direction and type of DCH. The interface generates the DATA CLOCK signal used for clocking the Nova ports' input registers, the three discrete control signals used for selecting the proper Nova port, and the DEVICE COMPLETE signal used for setting the Nova DONE flag. The interface also supports single operation 16-bit transfers should this capability be added to the Cromemco at a later date. Therefore, the interface completely supports the new IEEE S-100 bus standard (see Appendix A). Unfortunately, there were no pre-wired circuits that could implement this specialized interface; therefore, it had to be designed, fabricated, and tested in-house.

This interface operates differently when it is sending data to the Nova than it does when it is receiving data. When sending data the most significant byte of the word to be transferred is stored in a latch until the least significant byte can be sent. At the time that the least significant byte is transferred, the most significant byte, due to the latch, is already available on the upper portion of the data transfer path, causing a complete 16-bit word to be transferred to the Nova. The order in which the two bytes are sent to the Nova port is extremely important because it is during the transfer of the least significant

byte that all sixteen bits are clocked into the respective Nova input register by the DATA CLOCK signal. Also, the DEVICE COMPLETE signal, which is usually generated simultaneously with the DATA CLOCK, clears the Nova BUSY flag and sets the Nova DONE flag. The circuitry for receiving 16-bit data from the Nova is much simpler. Since the Nova has the data latched into one of it· ɔutput registers, there is no requirement for the Cromemco to latch this data. It is available any time the Cromemco wishes to read it. Therefore, the Cromemco merely inputs the most significant byte in one operation and the least significant byte in the next.

The Cromemco uses six of its I/O port addresses to select the proper byte of the Nova's A, B, or C Port registers. Generally, inputting from or outputting to a Cromemco I/O port determines whether the Nova port's output or input register, respectively, is selected. The I/O port assignments were made with great care to insure that they would not conflict with assignments already established for the standard Cromemco Z-2 system and also to provide an easy association to their Nova counterparts during the software writing phase of the project (see Appendix B). Additionally, the memory chosen for the Cromemco system is capable of operating under the IEEE standard for the S-100 bus which allows single operation 16-bit memory accesses and data transfers. Unfortunately, the Cromemco system only supports 8-bit transfers, but to allow for potential system

improvements utilizing a 16-bit microprocessor and/or a 16-bit DMA controller, the pairs of 8-bit Cromemco I/O ports used to communicate with each of the Nova 16-bit ports had to be mapped contiguously. With these criteria in mind, the I/O ports were assigned in pairs so that the Cromemco I/O ports $A0 - $A1, $B0 - $B1, and $C0 - $C1 corresponded to the low and high bytes of the registers of the Nova A, B, and C Ports, respectively. Hardware was also added to provide the 16-bit data transfer potential. The sixteen request (sXTRQ) and the sixteen acknowledge (SIXTN) signals, as required by the IEEE standard, were implemented to allow the pairs of contiguous I/O ports to be accessed as if they were a single 16-bit port.

To illustrate the technique implemented by the interface for transferring 16-bit data words to the Nova, suppose the Nova's C Port input register is to receive data. The Cromemco would first output the most significant byte of the data word to its $C1 output port. This operation would cause the Cromemco to latch this byte of data on the most significant byte of the data transfer path between the two systems. Next, it would output the least significant byte of the data word to its $C0 output port. This places the least significant byte on the data transfer path so that now the entire 16-bit word is present on the data path. The output to $C0 also generates the C SELECT signal which allows the DATA CLOCK signal to clock the entire sixteen

bits of data into the Nova's C Port input register. In addition, the DEVICE COMPLETE signal clears and sets the Nova BUSY and DONE flags, respectively, completing the transaction. When data is to be transferred from the Nova to the Cromemco, the transaction is much simpler. For a transfer of this type, the Cromemco would simply accomplish an input from its $C1 input port by generating the C SELECT signal. This would cause the entire 16-bit data word contained in the output register of the Nova's C Port to be gated on the data transfer path, but only the most significant byte of this data would be gated onto the Cromemco's data bus. An input from the $C0 input port would cause the same data word to again be placed on the data transfer path, but now the least significant byte would be placed on the Cromemco's data bus. It should be noted that the order in which two bytes of data are <u>input</u> from the Nova is inconsequential.

For programmed I/O handshaking and DCH control purposes, three additional I/O ports were assigned in the Cromemco system. The Nova BUSY and DONE flags are accessed by reading port $D0. The Cromemco's DCH/DMA control register is loaded by an output to port $D0. This control register has two bits which activate the Nova DCH control lines. Another bit of this control register allows the DEVICE COMPLETE signal to be disabled for all transfers from the Cromemco to the Nova A Port. DCH requests are generated by an output to port $C2. The DCH request can only be

generated if the proper DCH activation bit is set in the Cromemco DCH/DMA control register. The DCH/DMA control register has several bits that have been reserved for control of Cromemco DMA controller circuitry that is to be added at a later date.

I/O Channel Software and Protocol

The software for the I/O Channel was developed in two stages. The first stage provided a quickly constructed crude capability for digitizing speech and for regenerating the digitized speech to support the other theses currently in progress. During the second stage the software to allow the Nova to use the Cromemco as an I/O channel was developed. The second stage software duplicated some of the first stage capability, but the capability provided by the first stage was heavily used and therefore, proved to be very worth while.

The original requirements to be met for speech I/O were a sample and regeneration rate of 10 KHz and storage of the speech data on the Nova's disk. During the first stage two routines were written in the Cromemco and two companion routines were written in the Nova to accomplish this. Each pair of routines, one in the Nova and one in the Cromemco, provided for speech data to be transferred in only one direction. The two Cromemco routines were known as D2A8K which regenerated the speech and A2D8K which sampled the

speech. These routines set up the timers and interrupt control circuits on the CPU Support Card to interrupt the Cromemco processor at the desired rate and also included the interrupt service routine which initiated the next conversion cycle. In addition, the routines transferred the data between the Cromemco memory and the converters and between the Cromemco memory and the Nova. In order to store as much speech data in the Cromemco's memory as possible, it was decided to reduce the sample rate to 8 KHz. This rate allowed approximately three seconds of acceptable quality digitized speech to be contained in the Cromemco's memory before an additional transfer to or from the Nova was required. This quantity and quality of data proved to be very acceptable for supporting the other theses.

The two programs written for the Nova were called D2A, which transferred data from the Nova to the Cromemco and A2D which transferred data the other way. The initial approach for transferring the data was to temporarily buffer it in the Cromemco until it could be transferred to the Nova. Once the data was in the Nova, it was temporarily buffered until it could be put on the disk. Since this was to be done in real time, the software was written so that if the Nova or Cromemco fell behind while passing data, the transfer would be aborted. It was quickly established that the Nova was unable to keep up, so another approach had to be taken.

The software was originally written to store the speech data on the disk in random files, since this was the quickest approach. It was rewritten to use contiguous files, but the Nova was still unable to keep up. The only other change which could be made and still keep the transfer in real time was to use the DCH, since the data was currently being transferred by programmed I/O. Since there was insufficient time available to support this time consuming development, another method of transferring the data had to be derived. It was decided to sacrifice the real time transfer capability and sample as much data as the Cromemco's memory could hold before transferring it. This allowed the data transfer to take as long as required without interfering with the sampling or regeneration process in the Cromemco. Approaching the problem in this manner greatly simplified the software required for the Nova, further reducing the time required to get the system operational. The main programs in the Nova were written in Fortran; however, the software to accomplish the actual transfer between the two systems was written in assembly language and called from the Fortran program.

The ideal way for the Nova to communicate with the Cromemco is through a modification to the Nova operating system. Such a modification would make the Nova aware of the special device requirements of the Cromemco, but to accomplish this modification, the appropriate drivers would have to be written in the Nova assembly language and

properly linked to the I/O portion of the operating system. The advantage provided by a modification of this type is the ability of Nova programs to communicate directly with the Cromemco without the need of an intermediate interfacing subroutine. Unfortunately, it was found to be very difficult to make this type of modification to the Nova operating system since it has very little documentation on its actual internal structure and time constraints were a significant factor. Therefore, intermediate Nova assembly language routines were exclusively chosen to provide the actual data communication capability with the Cromemco.

While the first stage of software development provided considerable insight into the limitations and capabilities of the hardware and software of both systems, the second stage involved creation of the actual I/O CHannel Operating System (CHOPS) (see Appendix C). The CHOPS enables the Cromemco to function as a programmable I/O channel for the Nova. Although the CHOPS resides in the Cromemco, it operates in conjunction with a special Fortran subroutine called CHANNEL that resides in the Nova (see Appendix D). CHANNEL has the responsibility of insuring that Nova programs using the I/O channel observe the communication protocol between the two systems and allows the Nova to invoke tasks in the Cromemco. To insure that the CHOPS, CHANNEL and their communication protocol have no adverse effect upon the normal capabilities of either system, the constraints of the

hardware and software of both systems played a major role in their creation. The I/O Channel communication protocol, outlined in Appendix E, was created specifically for implementing the required I/O channel capability; therefore, its format and structure may only apply to this application.

For the I/O channel to operate in the required manner, it was necessary for the Cromemco to become an intelligent device for the Nova. To provide this intelligence, Cromemco software tasks which support the data and processing requirements of the Nova had to be generated. However, the Cromemco must also appear as a normal device to the Nova. In order to accomplish this, an I/O channel communication protocol was established to provide a standardized method for the Nova and Cromemco systems to communicate. Under this protocol, the Nova CHANNEL subroutine sends commands and command parameters to the Cromemco. The Cromemco acts on these commands by invoking the required tasks and transferring data to and from the Nova.

CHANNEL is a compact Fortran subroutine with several assembly language drivers that must be called from a Fortran main program. The main program passes the data direction, the command mode, the parameter count, and the task identification as calling arguments to the subroutine which uses them to form the Cromemco command. When data and/or task parameters are to be sent to the Cromemco, they are passed in their respective arrays. All data received

from the Cromemco is returned in the data array. If a DCH is involved, the file specified by the calling program is used to transfer the data. The name of this file must be passed to CHANNEL in a character array. The subroutine properly formats and formulates the information that it is passed, and then begins to communicate with the Cromemco under the established protocol. All errors it detects, including those identified by the Cromemco, are returned, as an argument, to the calling program by the subroutine. CHANNEL implements the full requirements of the protocol; therefore, it allows the addition of new tasks via Cromemco software or hardware changes, without requiring changes to itself.

The CHOPS was written entirely in the Z-80 assembly language, and was designed and developed for the Cromemco to implement the required I/O channel protocol. Under this protocol, CHOPS receives commands and parameters, validates this information, then accomplishes the commanded task. Command words can be accompanied by as many as sixteen parameter words. Parameter words allow such things as data block size, channel number, frequency, duration, speed, scale factor, or any other special data that a task may require. One of four modes is always associated with each command. Two of these modes involve data transfers. One of these two indicates a transfer by programmed I/O while the other indicates a transfer using the DCH. A bit in the command word indicates the direction that data is to be

transferred. One of the two remaining modes is used to indicate that no data is to be transferred, while the other indicates that an active command should be aborted or cancelled. The command abort mode is always associated with a specific command. This allows the Nova to cancel active commands and is very important if a multi-tasking capability were to be added to the CHOPS in the future. These command modes allow the Nova to set up for a data transfer without being concerned with the command itself. The foremost advantage of the CHOPS and the channel protocol is that new commands can be added to the system without the necessity of changing the Nova's software.

There are three types of 16-bit words that are defined by the CHOPS protocol (see Figure 1). Two of these words, the command word and the parameter word, are used to control the Cromemco system's operation. Under the current configuration, these two types of words can only be originated by the Nova and are sent to the Cromemco to cause it to begin the execution of a particular task. The third type of word defined by the protocol is the data word.

The command word is composed of six fields. These fields, beginning at the most significant bit and moving toward the least significant bit of a 16-bit word, are identified as the Command/Parameter (C/P), Task Identification (TID), Direction/Error (D/E), Command Status (CS), Command Mode (CM), and Parameter Count/Error Code (P/E)

COMMAND WORD

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
C/P	TASK ID							D/E	CS	MODE		P/E			

Bit 0 = Command/Parameter ID Field (C/P)
1 = Command word
0 = Parameter word

Bits 1-7 = TASK ID Field (TID)

Bit 8 = Direction/Error Field (D/E)
- if in original command from Nova
1 = Cromemco to input from Nova or output to peripheral
0 = Cromemco to output to Nova or input from peripheral
- if in echoed command from Cromemco
1 = Error code in P/E field
0 = No error occurred

Bit 9 = Command Status Field (CS)
1 = Command complete
0 = Command in progress

Bits 10-11 = Command Mode Field (CM)
00 = Non data transfer
01 = Programmed I/O data transfer
10 = DCH block data transfer
11 = Abort task

Bits 12-15 = Parameter Count/Error Code Field (P/E)

PARAMETER WORD

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
C/P	PARAMETER VALUE														

Bit 0 = Command/Parameter ID Field (C/P)
1 = Command word
0 = Parameter word

Bits 1-15 = Parameter Value

DATA WORD

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
DATA VALUE															

Bits 0-15 = Data value

Figure 1. I/O Channel Communication Words

fields. The C/P field is composed of one bit which is used to identify command words from parameter words. This bit will always be set for a command word. The TID field is a 7-bit field that uniquely identifies each task. The D/E field is another one-bit field and is used by the Nova to tell the Cromemco which direction the data associated with a command, if any, is to be transferred. When this bit is set, data will be transferred from the Nova to the Cromemco. When it is cleared, data will be transferred from the Cromemco to the Nova. The Cromemco always echos commands received from the Nova. If this bit is set in an echoed command word, it indicates that an error has occurred in the Cromemco. The Nova can then extract the code for this error from the P/E field of the command word, "OR" it with its own error code, and return this composite error to the calling program. The CS field is also a one-bit field that indicates the status of a command. This bit is set by the Cromemco only at the completion of a command and serves as a flag to the Nova to indicate that this is the last response that the Cromemco will send with respect to this command. All responses from the Cromemco with this bit cleared indicate that the command is still in progress. The CM field is an encoded field composed of two bits. This field tells the Cromemco the mode of the command that the Nova is invoking. These two bits are encoded to represent four different modes. The bit pattern "00" indicates a command that requires no data to be transferred. A "01" pattern

indicates that data associated with this command will be transferred using the programmed I/O capability of the interface. A pattern of "10" indicates that the commanded data will be transferred using the DMA/DCH capability of the interface. A pattern of "11" indicates that the Nova wants the Cromemco to abort the command represented in the TID field. The last field of the command is the 4-bit P/E field. When a command is sent from the Nova, this field contains the parameter count for the number of parameter words that will follow this command. Since this field is comprised of only four bits and a count of zero has been assigned to indicate that no parameters will follow the command, the maximum parameter count it can contain is fifteen. However, mode "01" (programmed data I/O) or mode "10" (DCH data I/O) commands must have a data count or block size, respectively, associated with them. This means that these two modes can have as many as sixteen parameters even though the P/E field only allows fifteen. The extra parameter will always immediately follow the command word. The P/E field may also be loaded with an error code by the Cromemco. When the Cromemco places an error code in this field, it sets the bit of the D/E field to indicate to the Nova that an error has occurred.

The parameter word is comprised of two fields. The most significant bit is the C/P field, which is always a zero for parameter words. The remaining fifteen bits make

up the Parameter Data (PD) field and are used for transferring the additional information that may be required by the Cromemco to properly process the command.

The data word is used to transfer the actual collected or processed data between the systems. The data word is allowed to be a full 16-bit word so that the Cromemco could, if necessary, accomplish pre- or post-scaling of any of the data that it handles. This feature can relieve the Nova from having to reconstruct or format data of smaller word sizes than it uses in its normal operations.

Under this protocol the Cromemco is required to handshake with the Nova when command, parameter, or data words are transferred using programmed I/O. This requirement allows the Nova to remain in complete control of the Cromemco, since it can monitor the result of each transaction, and intercede with a new command if errors are detected. The handshake is accomplished by the Cromemco echoing the command word that caused the transaction to take place. By monitoring the D/E, CS, and P/E fields of the echoed command, the Nova can determine the status of the last transaction. One further requirement placed on the Cromemco during parameter transfers is to decrement the P/E field after each parameter is received. The only exception is that the Cromemco does not decrement the P/E fields after the data word/block size is transferred during mode "01" or "10" commands.

The way in which the Cromemco handles a command primarily depends upon the command itself. However, in order to begin the actual process or task that the command is requesting, there are protocol requirements that it must observe. These particular requirements depend mainly upon the mode of the command, whether parameters are to be sent with the command, and the direction that data, if any, is flowing. The channel protocol requires that the command sequence begin by the Nova placing a command word in its C Port output register and setting its BUSY flag. Upon detecting the set BUSY flag, the Cromemco reads the Nova's C Port output register, checks the command for validity, insures that the CS bit of the command word is cleared, and echos the command by writing it into the Nova's C Port input register which completes the handshake for the command word transfer.

For a mode "00" command without parameters, the Cromemco will begin executing the commanded task immediately after the command word has been echoed to the Nova. If parameters were included with the command, the Cromemco would collect and validate the parameters before attempting to execute the command. The Nova must clear the DONE flag, which is set as a result of each echo. This will allow the Cromemco to echo the original command with the D/E and CS bits set when an intermediate error is detected during the command execution. Since the Cromemco waits for the DONE flag that was set by a previous echo to clear before

transmitting again, the channel could become hung if the Nova does not clear the DONE flag after each echo reception. The P/E field will also contain a unique error code identifying the error. Normally, as a result of an error, the Cromemco will completely abort the commanded task, return to its command collection mode and must be re-commanded to again invoke this task. Hopefully, as more elaborate tasking software with built-in error correction schemes evolves, this can become the exception rather than the rule. When a commanded task completes the original command without an error, the D/E bit will be cleared and CS bit will be set in the original command word, then it is echoed to the Nova. The Cromemco then returns to its command collection mode to await the next command.

When a mode "01" or "10" command is received, the Cromemco must insure that it first collects the extra word count/block size parameter and then any other parameters that are indicated by the count contained in the command word P/E field. It is the responsibility of the commanded task to accomplish the actual data transfer. These tasks can call CHOPS routines which have been designed specifically for this purpose. Utilization of these routines insures that the I/O channel protocol is observed. However, special high data rate applications may require these tasks to provide their own specialized data transfer routines compatible to both the tasks requirements and the

channel protocol. Data transfer is accomplished via the programmed I/O or the DCH capabilities. During a mode "01", programmed I/O data transfer, the Nova sends the Cromemco a special data command, which has been assigned a TID of $7F, to indicate that the next word will be data. The Cromemco must validate the command, and echo it with the D/E and CS bits cleared. It must then transfer the data and again echo the data command, if an error was not detected, with its D/E bit cleared and its CS bit set. If either the Data Command or the data word following it is in error, the D/E and CS bits will both be set and the respective error code will be placed in the P/E field of the echoed Data Command. This Data Command from the Nova, Data Command echo from the Cromemco, data word transfer, and Data Command echo from the Cromemco sequence is repeated for each data word that is transferred. After all the data has been transferred and all other processes imposed by the command have been completed, the Cromemco will echo the original command with the CS bit set and return to its command collection mode.

For a mode "10" DCH transfer, the block size is received as the extra parameter. This block size is in increments of 256 words and is used to indicate the number of data words that the Cromemco is to transfer before sending the Data Command to request the next block. For each block transfer, the Nova indicates that the block buffer is ready by clearing the DONE flag <u>without</u> setting BUSY. The Cromemco then continues the transfer, block by block, until

the Nova places the Data Command with the CS bit set in its C Port and sets the BUSY flag. The Cromemco validates this Data Command, completes the remaining tasks implied by the original command and echos the original command with the appropriate bits set as in the previously discussed modes. The Cromemco can also terminate the DCH transfer by setting the *DONE flag. This is accomplished by loading the Nova's* C Port with a Data Command that has its CS bit set.

The "11" mode allows the Nova to cancel or abort a task that is in operation. All parameter conventions and echo hand shaking, as previously discussed, still apply to this command. Since the Nova does not interrupt the Cromemco, the Cromemco must be looking for input from the Nova in order to receive it. This is accomplished during task execution by periodically checking to see if the Nova has sent a new command and by testing all commanded requests from the Nova for an abort. When detected, the Cromemco services this command by merely deactivating the task in progress and echoing the abort command word with its CS bit set.

As can be seen, the I/O Channel is quite versatile. It and its protocol have been designed to support multi-tasking in the event that the necessary additional software and hardware to support such a capability is added to the I/O channel in the future.

III. Design and Fabrication

This section examines the design of the Nova's GPI board and the interface board in the Cromemco system and how they work together. It also describes the software developed on both the Nova and the Cromemco to implement the initial speech I/O capability as well as the data channel which came later.

The Nova Interface Board

The GPI is built by GD to provide an interface between the Nova's I/O bus and the peripherals. It was used as a basis for the hardware on the Nova's side of the data channel. About two-thirds of the board is prewired. The remaining third, located along the edge of the board away from the backplane, is the breadboard area. Two rows of wire-wrap pins separate the prewired area from the breadboard area of the board. These pins are connected to the various inputs and outputs of the prewired circuits that are ma- -vailable to the designer. To allow enough clearance for a board to be placed above the GPI, the wire-wrap pins are quite short. Therefore, they can only hold about two wraps.

The GPI cannot be used as it comes from the factory. It must be configured by the user for his particular application by interconnecting the wire-wrap pins. Although this provides a great deal of flexability; DG does not

supply very specific information about how to accomplish this. The information used to configure the board was derived indirectly by examining DG's description of the I/O bus. Most of this information is contained in the following pages. Before attempting to discuss the registers, it should be noted that the bit number of the Nova's data bus is reversed from the numbering used by most computer manufacturers. For the Nova, the most significant bit is denoted as DATA0, while the least significant bit is denoted as DATA15.

Each programmed I/O instruction includes the six-bit device code which designates the particular interface to be involved in the operation. During the execution of the instruction, the device select lines of the I/O bus, DS0# through DS5#, carry the device code of the desired interface. Note: a pound sign (#) on the end of the name of a signal indicates that it is active low. In addition to the device select lines there are three control lines on the I/O bus (DATIA, DATIB, and DATIC) which designate an input to Port A, B, or C, and three lines, DATOA, DATOB, and DATOC, which designate an output to Port A, B, or C. The timing of these signals is shown in Figure 2. All six of the I/O signals are made available to the user through wire-wrap pins.

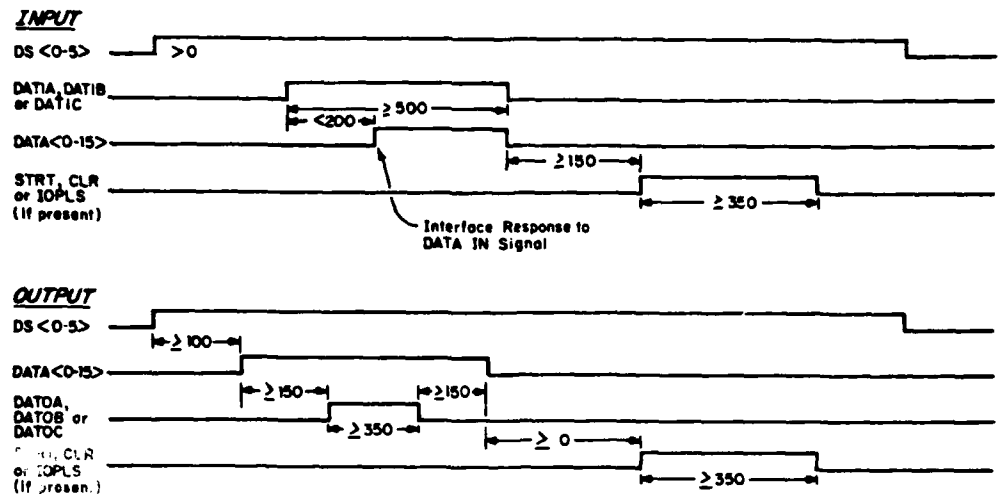


Figure 2. Nova Programmed I/O Timing Diagram

The GPI provides circuitry which can be wired to decode the device select bus for any one of the desired device code (see Figure 3). This circuitry is simply an eight input NAND gate with some inverters. It uses jumpers to feed either an inverted or noninverted signal from the device select bus to the NAND gate. The output of the NAND gate is buffered by two inverters to become DEVICE SELECT 1 and 2. The former is NANDed with each of the six input/output signals used by the ports to generate an active low signal that can be used to clock data onto or off of the Nova's data bus (see Figure 3). DATA OUT A#, for example, is generated at device 25 as a result of the DOA 25 instruction. This DEVICE SELECT signal is the primary signal that allows the Nova to communicate with the input and output registers of a device.

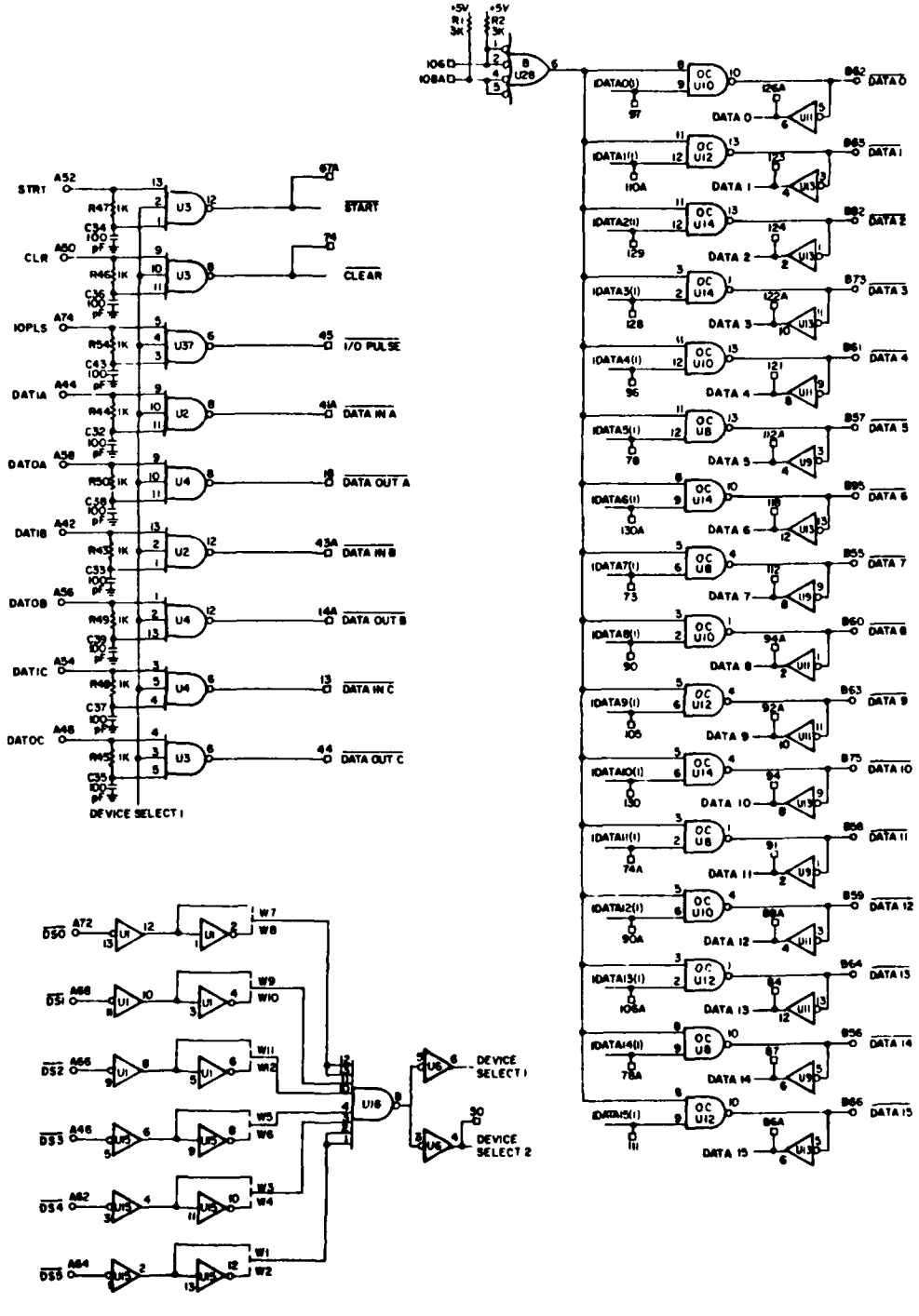


Figure 3. GPI Programmed I/O Control Signals and Data Bus Buffer

The input and output registers used for the A Port in this interface were prewired on the GPI (see Figure 4). They each consist of four 4-bit parallel-access shift registers (SN74195). Each register is wired the same way except for the shift/load line and the data inputs and outputs. The shift/load line of the output register provides for the ANDing of two inputs to it, while the input register does not. The clock inputs of both registers will NAND two signals. A serial input and a clear input are also available to the user. The data inputs of the output register are connected to the Nova's data bus through inverters. The inverters both buffer the data bus and transform the data from active low to active high. Unfortunately, the data outputs of this register are also buffered through inverters resulting in complimentary data. The inputs of the input register are connected to wire-wrap pins which are connected to the 16-bit input data path of the interface cable. The outputs of the register are gated on to the Nova's data bus through open collector NAND (SN7438) gates (see Figure 3). This both buffers and compliments the register's output as required by the bus. The registers were assigned to the A Port by tying DATA OUT A# (pin 18 on the GPI) to the input register's clock (pin 33) and by tying DATA IN A# (pin 41A) to one side of the gates on the outputs of the output register. Pin 17 and pin 32 of both registers are tied low to place them in the parallel load mode.

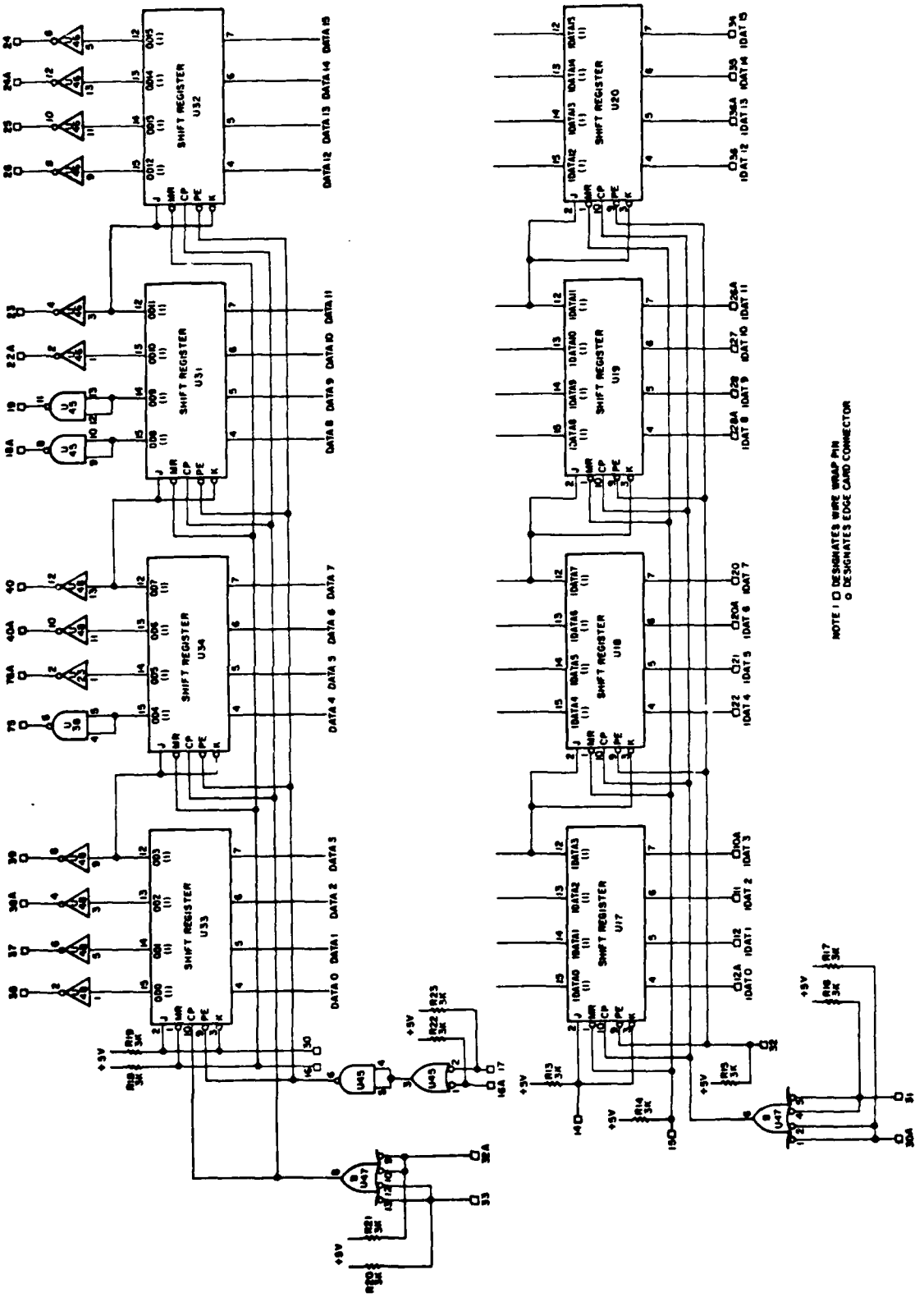


Figure 4. GPI Input and Output Data Registers

The prewired register on the GPI designed to be used as an address register for DCH transfer was assigned to Port B (see Figure 4). This register consists of four 4-bit binary counters (SN74177) which are configured to increment from 0000 to FFFF (Hex) when CL1 of the counter is clocked. The counters can be parallel loaded so the register can be written to by the Nova processor. The outputs of this register are connected to the data bus through open collector NAND gates and the inputs are connected to the bus through inverters the same way as the A Port registers. The CA CLOCK (connected to CL1), CA DATA (DATA STROBE of the counter), and the NAND gates on the outputs all allow the user to OR two signals together before supplying them to the register. The user is also given access to the reset line of the counter through a single input. Wire-wrap pins are connected to the output of this register but the inputs are not made available. The register is assigned to the B Port by tying DATA IN B# (pin 43A) to one input of the NAND gates (pin 108) and connecting DATA OUT B# (pin 14A) to CA DATA (pin 60). The required connections for the DCH will be discussed later in this section.

The last prewired register to be discussed is the word-count register. This register is used simply as an output register for the C Port since the Cromemco will track the word count internally. This register consists of four SN74177's configured the same way as the address register (see Figure 5). The main difference between the two is

that the outputs of the word count register are connected through inverters to wire-wrap pins. An output register for the C Port had to be built since there were no additional prewired registers on the GPI. The new output register had to have access to the Nova's data bus which, unfortunately, is not provided anywhere on the GPI. Therefore, an output buffer of NAND gates using four SN7438s was built in the breadboard area of the GPI and its outputs were wired to the circuit traces in the prewired section of the board which would connect it to the data bus. Four shift registers (SN74195) were added to the GPI to form the rest of the output register. All the inputs on one side of the NAND gates were tied together and connected through a buffer to DATA IN C# (pin 44). This assigned the new output register to the C Port.

In order to complete the description of the programmed I/O section of the board, the BUSY/DONE network must be discussed (see Figure 5). The BUSY/DONE network uses two flip-flops (SN7474) to provide the BUSY and DONE flags. The output of these flip-flops are connected to wire-wrap pins for user access and NANDed with DEVICE SELECT 2, generating the signals SELD# and SELB#. SELD# and SELB# are placed on the I/O bus when the device is selected so the processor can read the flags. The rest of the BUSY/DONE network provides the logic to perform the following functions:

1. Clear both flags when the I/O RESET# or CLEAR# signals are generated.

2. Set the BUSY flag and clear the DONE when the START# signal is generated.

3. Clear the BUSY flag and set the DONE on the positive transition of a clock applied to pin 56 (DEVICE ACKNOWLEDGE). DEVICE ACKNOWLEDGE has no effect if the BUSY is not set.

There are times that the I/O Channel protocol requires the Cromemco to set the DONE flag when the BUSY flag is not set; therefore, the BUSY/DONE network had to be modified. This was accomplished by lifting pin 12 of U22 (DONE flip-flop) from its printed circuit board connection and connecting it to +5 volts through a 3 k-ohm resistor. This modification allows the DONE flip-flop to be set each time DEVICE ACKNOWLEDGE is clocked. The clearing of the flag is not affected since CLEAR# uses the reset input of the flip-flop. The interrupt control logic is also associated with the BUSY/DONE network. Since interrupts were not implemented, they were disabled by tying the D input of the INTERRUPT DISABLE flip-flop, (pin 86) low.

Now the means of high speed, bulk transfer of data, the DCH, will be discussed. Since more functions must be performed for each DCH transfer, the interplay between the Nova's processor and the interface is more complex than that involved in programmed I/O. An interface requests a data channel by asserting the DCHR# line of the I/O bus. When the processor checks this line and finds it asserted, a DCH is executed. No software is required to perform the DCH itself; however, normal program execution is suspended

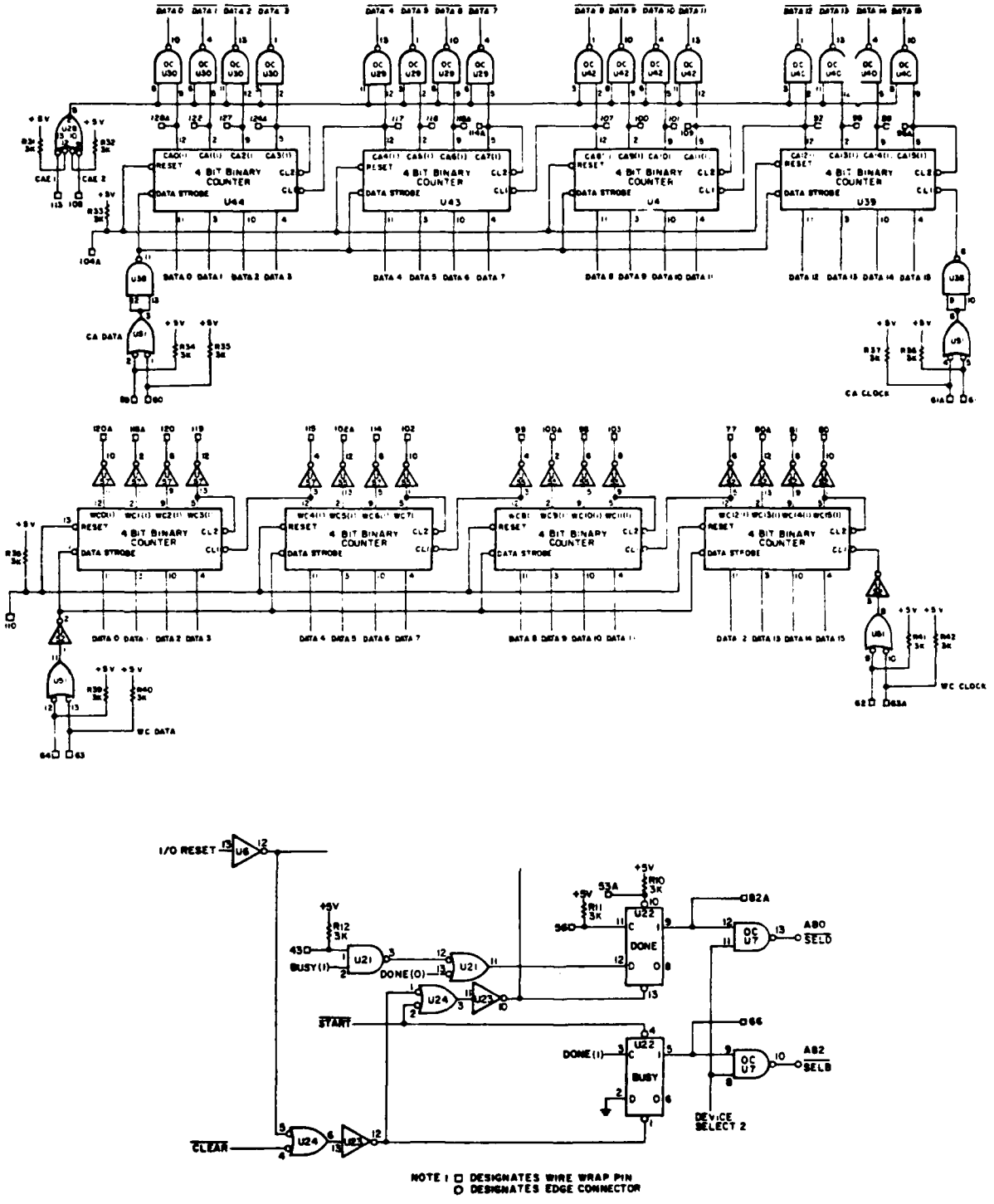
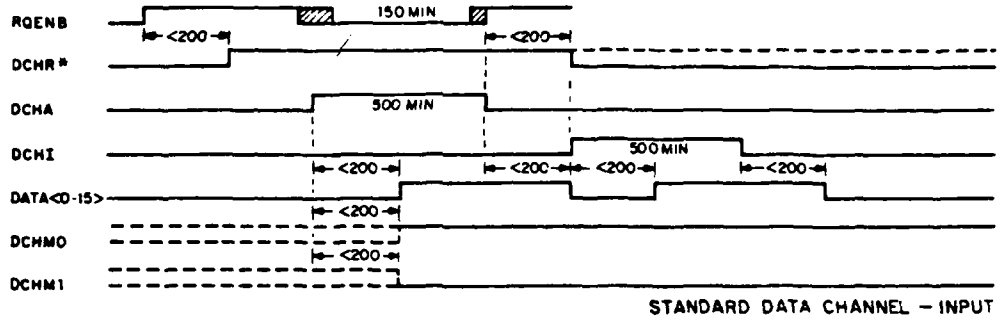
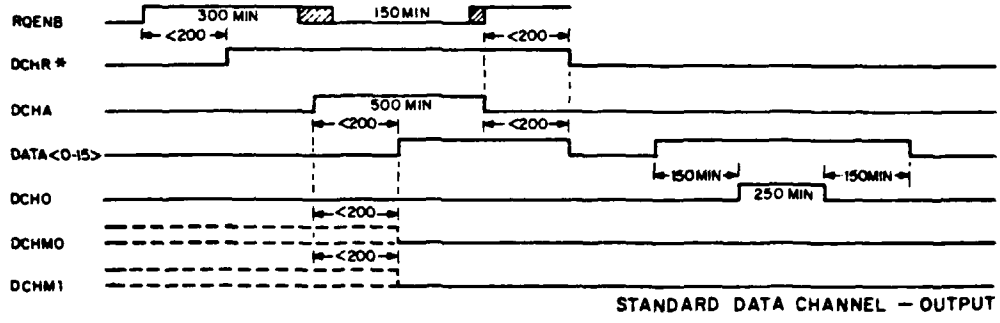


Figure 5. GPI Address and Word Count Registers and Busy/Done Network

during the transfer. This delay is not usually noticable since the transfer takes place in one to two microseconds. The signal RQENB# is used to sync DCHR# to the I/O bus (see Figure 6). The GPI uses the leading edge of REQENB# to clock DCHR# on to the bus. The processor issues the DCH acknowledge signal, DCHA# in response to the DCHR as the first step in a DCH transfer. This signal goes to all interfaces on the I/O bus with the processor expecting the memory address and mode of the transfer in response. A priority network rather than the device select bus is used to determine which interface is to respond to these signals. This priority network is accomplished by daisy chaining the interfaces with a signal which is called DCHP IN# as it enters and DCHP OUT# as it leaves. Under this system the interface, which is physically closer to the procesor on the priority chain, receives the highest priority. Normally each interface passes the DCHP# signal undisturbed on to the next interface; however, when this signal reaches the requesting interface, it is trapped so it does not pass to next interface (see Figure 7). A device is "selected" when it receives the DCHP IN signal in response to a DCH request.

Before one DCH cycle is complete, the DCH request from the interface being serviced must be cleared in order to prevent an immediate second transfer. Therefore the DCH SEL flip-flop (see Figure 7) is used to indicate that a DCH transfer is occurring for the interface. The DCH SEL flag





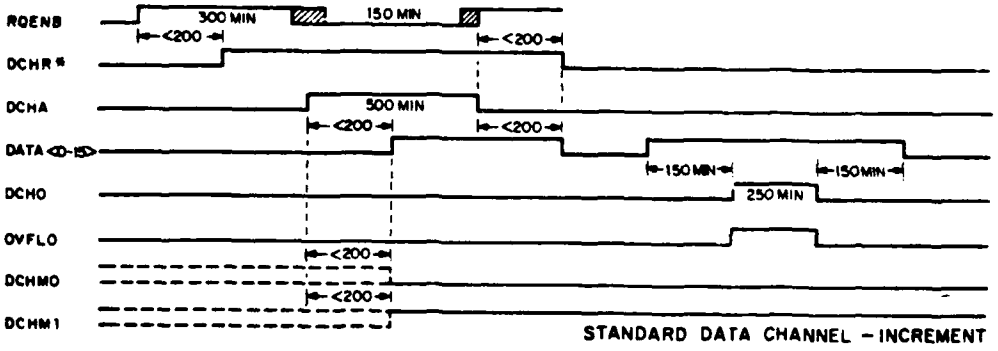


Figure 6. Nova Data Channel Timing Diagram

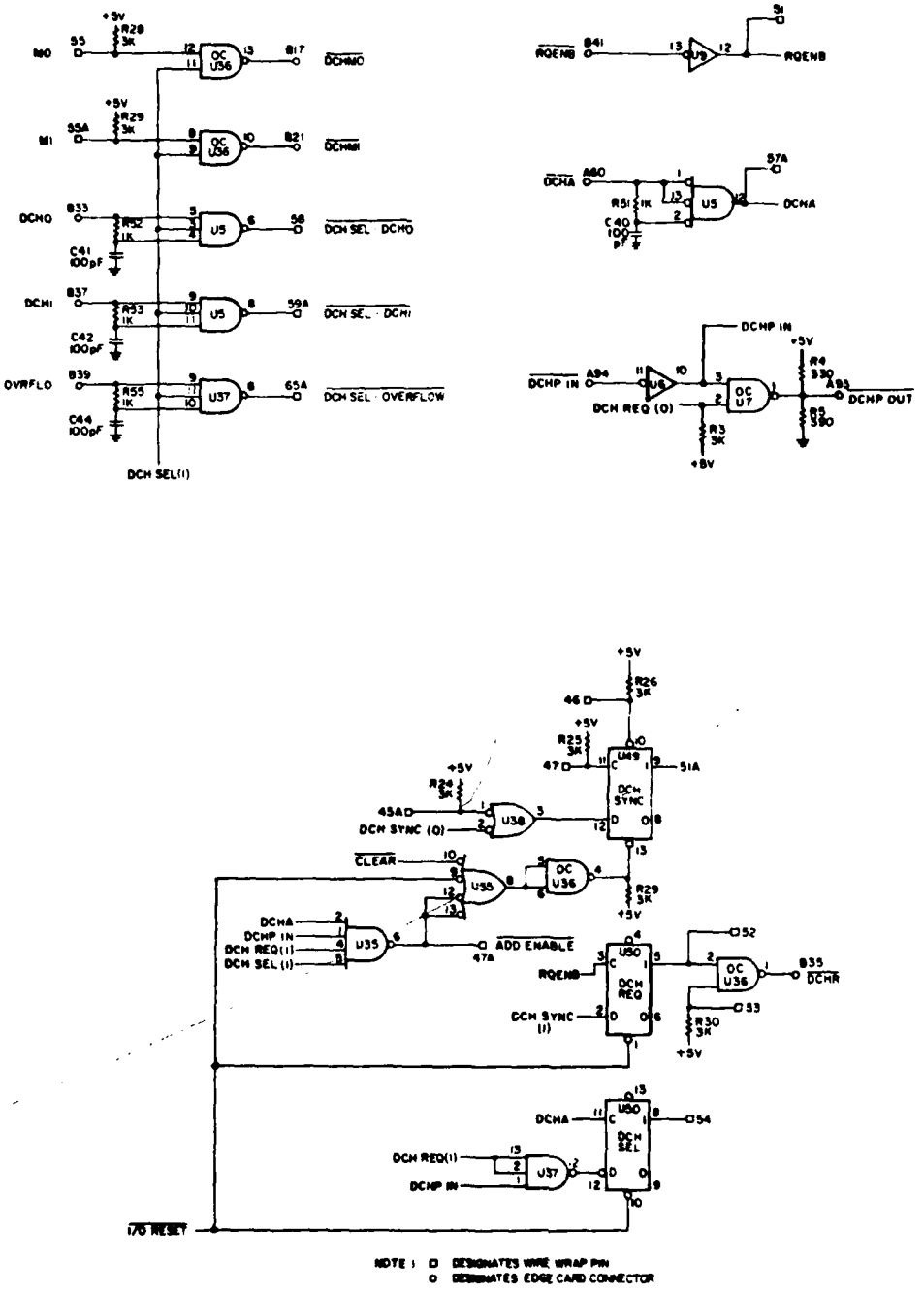


Figure 7. GPI Data Channel Control Logic

is set at the beginning of a DCH transfer for the interface and cleared at the beginning of the next. The DCH SEL is only set when DCHP is present. The DCH SEL flag serves the same purpose that DEV SEL does in parallel I/O. It indicates when the interface should respond to a transfer.

While receiving DCHA#, the interface with its DCH SEL flag set places the address contained in the address register (B Port) on the data bus. This is done on the GPI by NANDing DCHA, DCHP IN, the DCH REQ flag, and the DCH SEL flag (see Figure 7). The resultant signal is called ADD ENABLE#. This signal is also used to clear the DCH SYNC flag which then causes the DCH REQ flag to be cleared on the next REQENB. A DCH has four possible modes which are selected by asserting DCHM0# and DCHM1# on the I/O bus. The codes of the modes and their functions are "00" for Output, "01" for Increment, "10" for Input, and "11" for Add to Memory. DCHM0# serves as the most significant bit and DCHM1# the least significant bit of this code. Input and Output are with respect to the Nova processor and are self explanatory; however, Increment and Add to Memory need some explanation. Add to Memory causes a data word that is transferred from the interface to be added to the contents of the memory location supplied by the interface. The result of this addition is deposited into the memory location and also transferred back to the interface with a DCH out. The OVFLO signal on the bus will be pulsed if this sum is greater than $2^{16}-1$. Increment is similar to Output;

except, the contents of memory are incremented before being sent to the interface. The same value that is sent to the interface is also deposited into the memory location.

DCHI# and DCHO# are the I/O bus signals which indicate when the data should be placed onto or clocked off of the data bus, respectively. These signals are inverted and NANDed with DCH SEL by the Data Channel Control circuits to form DCH SEL*DCHI# (pin 58) and DCH SEL*DCHO# (pin 59A). To assign the A Port to the DCH, DCH SEL*DCHI# is connected to pin 106 of the input register (see Figure 4). The outputs of the register are gated onto the data bus on the low transition of this signal. In addition, DCH SEL*DCHO# is connected to pin 32A of the A Port output register (see Figure 4). The contents of the data bus are clocked into the register on the low transition of this signal.

The remaining circuits on the GPI were installed to multiplex the output registers onto the interface cable (see Figure 8). This was done with tri-state bus drivers. Inverting drivers (SN74240) were used for the prewired registers since their outputs are inverted by the GPI and noninverting drivers (SN74244) were used for the added register, C Port. One of three control lines (A SEL, B SEL, or C SEL) is used to select the desired output register. Each control line is connected, through a buffer, to the enable lines of the drivers for its respective register. These control lines are also used to select which register

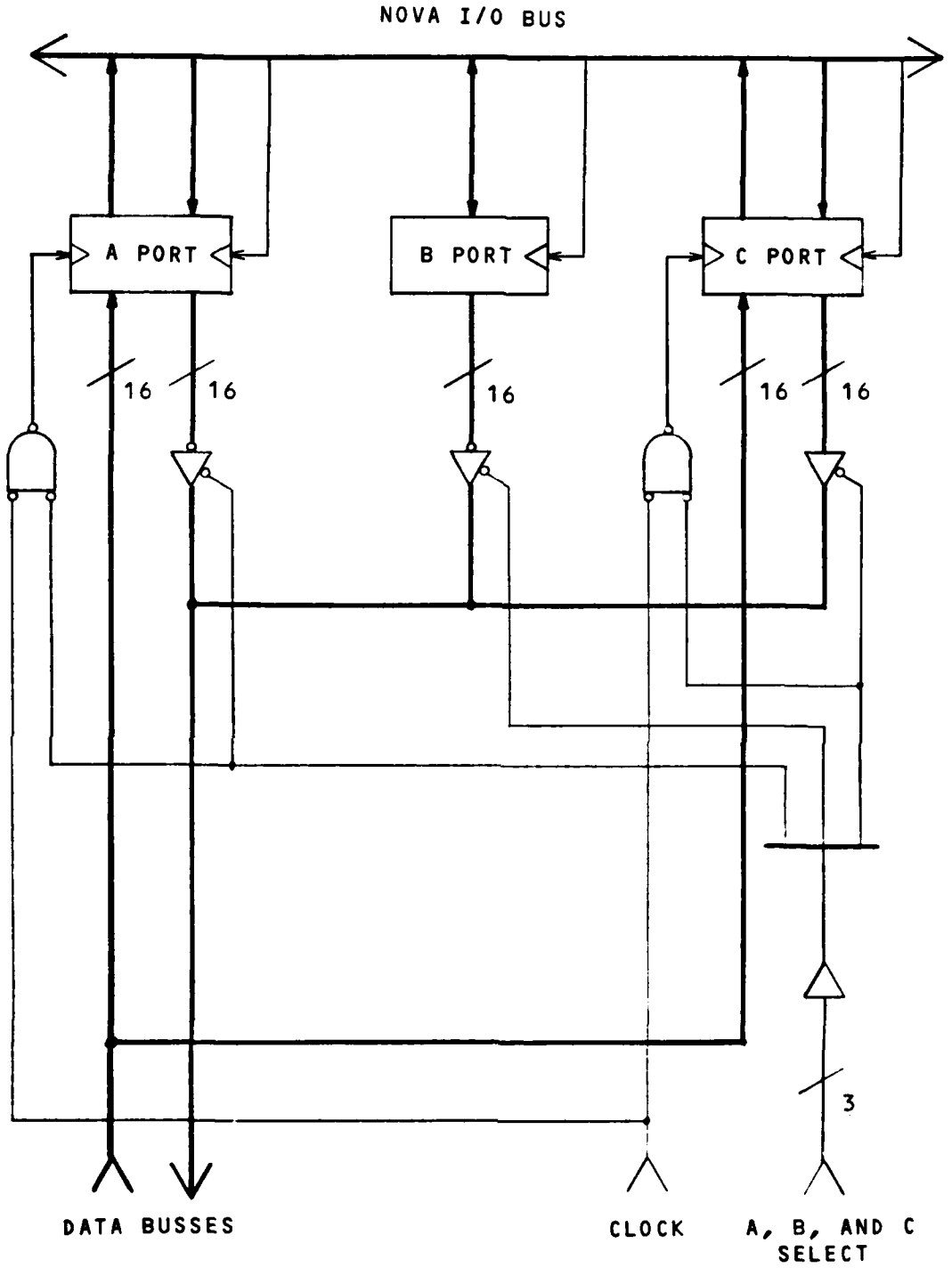


Figure 8. GPI Register/Data Path Interconnections

is to be loaded by the I/O Channel. A clock from the I/O Channel is directed to the appropriate register by its respective select signal. As indicated earlier, the B Port cannot be loaded and therefore is not clocked by the I/O Channel. However, a set of input and output registers could be put in the breadboard area which could function as both the address register during a DCH and normal input or output registers during programmed I/O.

Nova Software

The Nova software includes a Fortran subroutine called CHANNEL, and four Fortran callable assembly language subroutines, SANDS, CANDR, DCHTX, and DCHRX (see Appendix D). SANDS and CANDR communicate with the I/O Channel through the C Port while DCHTX and DCHRX handle any DCH transfers. As indicated earlier CHANNEL is designed to allow the addition of tasks to CHOPS without requiring changes to itself. It simply passes the number of the desired task on to CHOPS without decoding it. CHANNEL can still perform all the required data transfers by decoding at the direction and mode fields.

The task, direction of transfer, parameter count, and mode are some of the nine arguments passed by the calling routine to CHANNEL. They are first combined into a command word and then sent to the I/O Channel by CHANNEL. Each argument is checked to insure that it falls within the appropriate bounds. If the transfer is mode one, a data

count indicating the number of data words involved must also be passed. If the transfer is a mode two type, a block count must be passed. This block count is the number of Nova disk blocks (256 words each) which will be transferred at one time during a DCH. In other words this is the number of disk blocks which are collected in a buffer before being transferred by DCH. In addition, if the transfer is from the I/O Channel to the Nova and the user requires contiguous file to be created, the total number of blocks to be transferred must be passed. If a random file is required instead, a zero should be passed in this field. The name of the file which will serve as a source or destination of the data in a DCH transfer is passed in a character array. In addition, any parameters or data required for mode one are passed in PARRAY and DARRAY, respectively.

CHANNEL also provides considerable error detection (see Appendix F). Any error that is detected is returned as an argument to the main program. Since the I/O Channel can return errors of its own, in addition to the errors which occur within CHANNEL itself, the error word includes two fields. The most significant byte contains CHANNEL errors while the least significant byte contains errors returned by the I/O Channel. If an error occurs during the transfer of a parameter, PCOUNT will return with the number of the parameter being passed at that time. This also happens when data is transferred, except the number is returned in DCOUNT. If a Nova error occurs after a command has been

sent to the I/O Channel, CHANNEL will automatically transmit an abort command before returning to the main program. This insures that CHOPS is not left waiting for a particular response from the Nova. An abort is indicated by setting the most significant bit of CHANNEL's error byte. If the I/O Channel ignores the abort command, CHANNEL's error byte will be set to all ones indicating the I/O Channel is malfunctioning and should be reset.

Cromemco Interface Hardware

The hardware which interfaces the Cromemco to the Nova was constructed using a Universal Microcomputer Processor Plugboard (model #8800V-A) manufactured by Vector Electronic Company. This breadboard is designed specifically for the S-100 bus system and provides the S-100 edge connector and electrical traces for supplying power to the circuitry placed on the board. Pads are conveniently provided for installing up to four of the appropriate voltage regulators required by all S-100 bus interfaces. Also, the board is pre-drilled with component mounting holes on a one-tenth inch center pattern and provides an extensive ground plane. Wire-wrap sockets were installed on the breadboard to accommodate all the IC's used in this design. Although wire-wrap sockets are more expensive, the additional expense they incur is small when compared to the design evolution advantages and the interface maintainability gained through their use.

The actual interface circuit can be divided into three major functional areas. These consist of the I/O port decoding logic, the data path/data bus interface logic, and the DCH/DMA support logic. The I/O port decoding logic provides the necessary combinational logic to decode the I/O ports of both the Cromemco and the Nova by supplying the proper signals to the data path/data bus interface logic. The data path/data bus interface logic consists of several 8-bit tristate bus drivers and latches that are used to map the 8-bit data bus of the Cromemco into the 16-bit ports of the Nova. This logic also includes the circuitry that allows the Nova BUSY and DONE flags to be read by Cromemco. The DCH/DMA support logic is the part of the interface that accomplishes the required handshaking with the Nova's DCH circuitry. Provisions have been made in this part of the interface to include the necessary control logic for a future inhancement to the interface which will provide the Cromemco with a DMA capability.

The I/O port decoding logic derives its inputs from the signals propagating on the Cromemco/S-100 bus. These inputs consist of the least significant eight bits of the 16-bit address bus and the sINP, sOUT, pWR#, pDBIN, sXTRQ#, and, SIXTN# signals. When either the sINP or the sOUT signal is present on the bus, the output of IC3a (see Appendix G) becomes active, enabling the outputs of both of the four-line to sixteen-line decoders (SN74154), IC11 and IC12. IC11 is connected to decode the upper four bits of the least

significant 8-bits of the address bus. Its A, B, C, and D decoded outputs become active when an I/O port address of $AX, $BX, $CX, or $DX, respectively, appears on the low byte of the address bus during either an input (sINP) or output (sOUT) cycle. Likewise, IC12, the other decoder, is connected to the lower four bits of the address bus and the outputs that are used from it become active when an I/O port address of either $X0, $X1, or $X2 appears on the bus during the proper cycle. The decoded A, B, and C outputs from IC11 are combined by IC5a to generate a signal which indicates that access is required of a Cromemco I/O port having a counterpart in the Nova. All of these outputs are then gated with the bus read (pDBIN) and write (pWR#) signals by IC5b, IC5c, IC6a, IC6b, and IC6c to generate the necessary control signals to activate the circuitry of the data path/data bus control logic. The decoders also allow easy relocation of the ports on the interface should a conflict arise. This is because each decoder provides sixteen output signals for the four bits it decodes. Therefore, port relocation simply requires rewiring to the proper outputs of the two decoders.

The data path/data bus control logic maintains control with six 8-bit tristate non-inverting bus drivers (IC3, IC7, IC8, IC14, IC15, and IC17). These drivers are used to gate the least significant (IC3 or IC8 and IC17) or most significant (IC7 and IC15) portion of a 16-bit word to and

from the Cromemco data bus. When ports $A0-$A1, $B0-$B1, and $C0-$C1 are accessed by the Z-80, the proper drivers are activated to allow 8-bit bytes of data to be transferred. The circuitry composed of IC2b, IC2d, and IC10a allows the interface to respond to requests for 16-bit I/O transfers. This is accomplished by receiving the SIXTN# signal from the bus and allowing IC7 and IC8 to be activated to form a 16-bit input path or IC15 and IC17 to form a 16-bit output path. The SIXTN# signal is actually a request signal; therefore, it must be acknowledged. This is accomplished by generating the sXTRQ# signal at the output of IC2b. Unfortunately, the Z-80 processor does not support 16-bit memory or I/O transfers, but the memory in this system can. It was therefore decided to generate an interface to match the memory's capability in the event that the system is later upgraded to a 16-bit processor. One other bus driver, IC4, is included to allow the Nova BUSY and DONE flags and the other interface control flags to be placed on the Cromemco input data bus. It is activated when port $D0 is read. Additionally, IC16 provides an 8-bit tristate latch that is used for holding the most significant portion of each 16-bit word that is to be transferred out of the interface to the Nova.

A transfer from the Nova to the Cromemco takes place when a Nova port output register is selected via the A SELECT, B SELECT, or C SELECT signals originating from IC11. These signals also cause the signal from IC10d to allow

IC7 to become active, which places the most significant byte of the Nova register onto the Cromemco input data bus. Likewise the signal from IC2d enables IC3 to place the least significant byte of the selected Nova port ouput register onto the Cromemco input data bus. For transferring data from the Cromemco to the Nova, the most significant byte must be transferred first. It is latched by IC16 when the signal from IC6b is present. When the higher address of each port pair ($A1, $B1, or $C1) is written to, an output from IC6b is generated. Since IC17 is always enabled, the least significant byte will be placed onto the data path immediately when it is written via ports $A0, $B0, or $C0. Since the data in the latch is already present on the most significant half of the data path, the trailing edge (low to high transition) of the signal from IC6a, DEVICE COMPLETE#, causes sixteen bits of data to be clocked into the selected Nova register. To read the status flags the output of IC6c is activated by inputting from port $D0. This causes IC4 to place a byte containing the I/O Channel status flags, including the Nova BUSY and DONE flags, onto the Cromemco input data bus.

The DCH/DMA support logic is composed of IC21b, IC22a, IC25a, IC25b, IC26a, and IC26b. These devices generate the proper signals to handshake with the Nova's DCH control circuitry. The IC25a flip-flop is clocked by the signal from IC21b, which is generated by an output to port $C2.

The IC25a Q output then goes high and its Q# output goes low, generating a signal which is fed to an inverting open collector driver to generate the DCHREQ signal to the Nova. The Q# output of IC25a is also connected to the direct clear input of IC25b. This causes the Q output of IC25b (DCHDON) to go low each time a DCH request is generated. The Cromemco uses the low DCHDON signal to indicate that a DCH is in progress. When the Nova honors the DCHREQ signal it responds with it's DCHACK signal via IC26b. This causes IC25a to be direct cleared which removes the DCHREQ signal. The Nova responds with DCH SEL*DCHO# or DCH SEL*DCHI# after it has placed the appropriate data in its A Port. These two response signals indicate the end of the DCH cycle and are "OR"ed by IC26a to form a signal which clocks IC25b. Since the D input of IC25b is tied high, the Q output of IC25b also goes high generating the DCHDON signal. As a result of setting DCHDON, the circuit has completed its cycle and is ready for another request. The Cromemco must only initiate a new DCH cycle when the DCHDON flag is high. This flag is read by inputting from port $D0 and testing the next to the most significant bit. Writing to I/O port $D0 allows the direction of the DCH to be set and the inhibiting of the DEVICE COMPLETE signal when the contents of I/O port pair, $A0-$A1, are transferred to the Nova. The two most significant bits of output port $D0 are connected to the Nova's M0 and M1 inputs and set the direction for a DCH. A pattern of "11" sets the direction for input to the

Cromemco, while a pattern of "10" sets the direction for the Cromemco to output data. Bit 5 of this port, when cleared, will inhibit the Nova A Port input register clock signal from generating a DEVICE COMPLETE#, since this signal is used to indicate the completion of a DCH data block transfer.

The interface has been thoroughly tested, with the exception of the 16-bit transfer capability, and is now completely operational. The flexibility that was designed into its circuitry should accommodate most of the possible future enhancements to its capabilities.

Cromemco Software

The software which allows the Cromemco to communicate over the channel was written entirely in the Z-80 assembly language. Considerable thought was given to using a High Order Language, but the only two available for the Cromemco were Fortran and Basic. Neither of these seemed well suited for generating the required software. Although Fortran would not have been the optimum language to use, it would definitely have been a better choice than the interpretive Cromemco Basic. The inherent slowness of this Basic was made readily apparent after running several BASIC test routines when exercising the A/D and D/A converters for the first time. The major drawbacks of Fortran are that its bit manipulation capability is very limited and the memory requirements for the compiler generated code are difficult

to predict. If the Cromemco had the requirement of opening and closing disk files like the Nova, then access to the existing Cromemco operating system would have been necessary. This would have been much easier from Fortran and could possibly have reversed the decision to code in assembly language.

The assembly language of the Z-80 is probably one of the most robust of the existing 8-bit micro processors. The mnemonics are much easier to learn and employ than those of its predecessor, the 8080. The Z-80 will execute all of the instructions of the 8080, plus, almost as many again of its own. This richness of instructions made the Z-80 an ideal processor in which to implement the controlling software for this side of the channel.

The CHOPS is the software program that is resident in the Cromemco (see Appendix C). Its sole purpose is to provide a link between the hardware resources of the Cromemco and the processing power and disk storage of the Nova. The CHOPS has been written as a true remote operating system which remains completely transparent to the Nova system user, since the user communicates with it through the Nova's CHANNEL subroutine. On the other hand, the CHOPS does not itself execute the commands that it receives from the Nova. It only provides routines to transfer information between the Cromemco and Nova, observing the established comunication protocol. Therefore, it must have tasks

associated with each command coupled to it. These tasks can be written to implement any hardware capability in the Cromemco.

The CHOPS can be subdivided into five major areas consisting of the command/parameter collection and validation routine, the data transfer routines, the error handling routine, the tasks, and the command table (see Figure 9). The command/parameter collection and validation routine is responsible for listening via programmed I/O to either the Nova or an external development system for commands and their associated parameters. After receiving Nova commands and parameters, this routine does all the necessary error checking, then communicates the status of the commands and parameters to the Nova. If all are received properly, this routine will invoke the commanded task and, if the task is properly written, control of the Cromemco system will only be partially relinquished. The data transfer routines are responsible for observing all of the communication protocol required for programmed I/O or DCH transfers. They are invoked by a task to receive data for processing or output to a peripheral or to send data to the Nova that has been collected by a peripheral. The transfer technique and routines used for transferring data is determined by the Nova command and the actual task software. The error handling routines provides the CHOPS with the power to inform the Nova that an error or fault has occurred either during information transfer or during task execution. In its

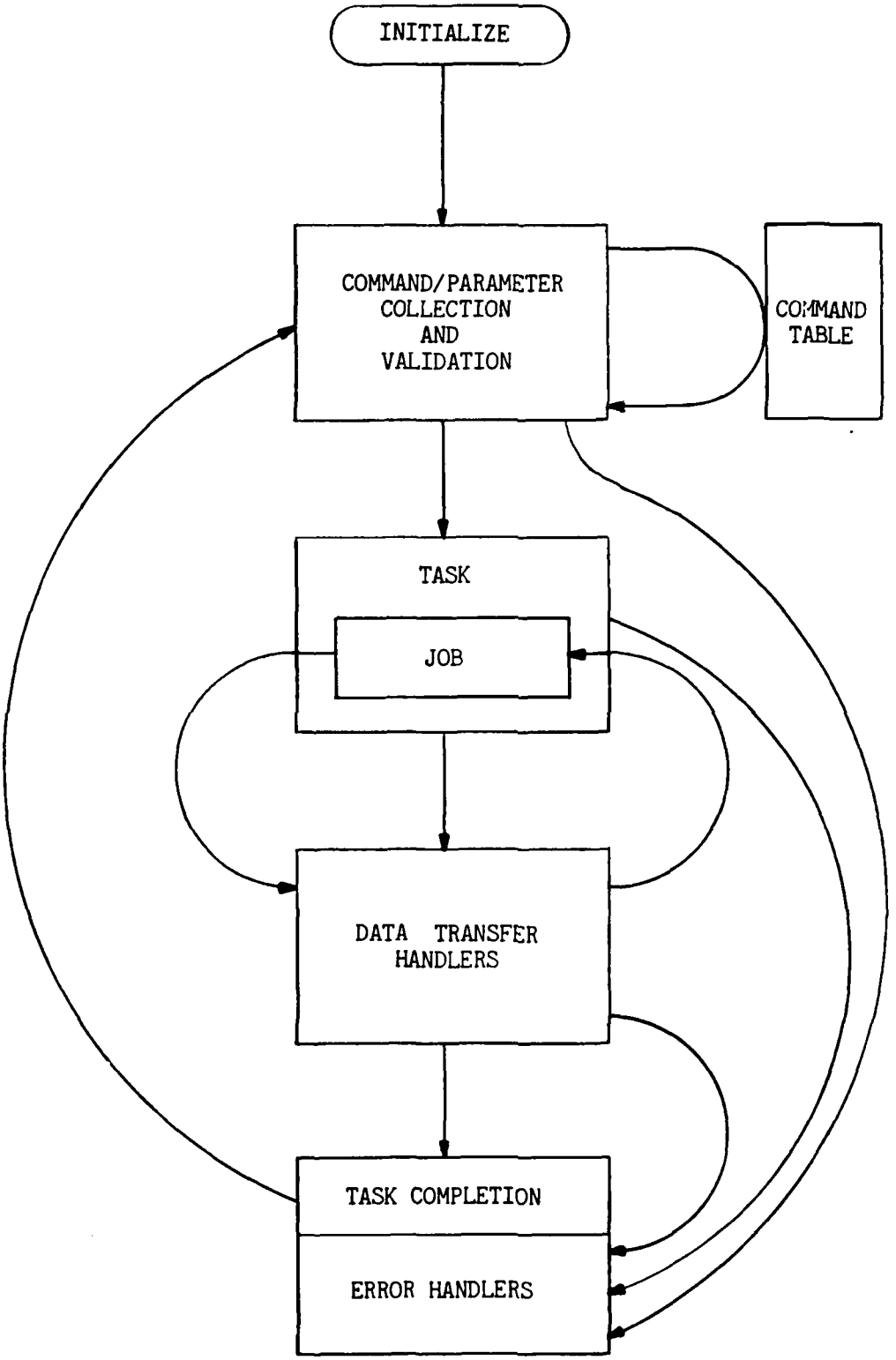


Figure 9. CHOPS Functional Block Diagram

present form, this routine automatically deactivates and terminates a task in which an error has occurred. The task routines provide the Cromemco, and consequently the Nova, with the ability to utilize resources contained within or connected to the Cromemco. Without the tasks, CHOPS would not know what to do with data it receives nor would it ever have any data to send to the Nova. The command table provides a link between the first three parts of the CHOPS and the tasks that may be added in the future. This table provides an address for the command information list. Each valid command has a list containing the number of parameters to be expected with the command, the bounds, both low and high, of these parameters and the address of the task routine itself.

If an instruction by instruction description of the CHOPS is required, the listings contained in Appendix C have been generously commented and should provide this detail. The following discussion deals primarily with what each routine is doing rather than how the routine does it. This information should be extremely beneficial and should be read first before attempting to understand the CHOPS at its Z-80 assembly instruction level.

The command/parameter collection and validation routine is the main protocol observer of the CHOPS. It is the routine that is in complete control of the Cromemco when CHOPS is first invoked. This routine first initializes the

command status table, prints a message to the Cromemco console screen and then waits for a command from either the Nova or the console keyboard. Currently, the console keyboard serves no real purpose. It is simulating a capability that a remote software development system will ultimately provide. When a command from the Nova is received, the command collection portion of the routine tests to insure that the most significant bit of the received word is set. This indicates that the word is indeed a command. If the most significant bit had not been set, the CHOPS would respond with an error and wait for the next command transmission. When a true command word is received, the most significant portion of the command and the direction bit are used to form an offset into the command table. This offset is added to the base address of the table and a 16-bit address is extracted from the table. For a command word to be valid, it must have a task routine available. This is signified by a non-zero most significant byte of the address extracted from the table. This address points to the command information list for the received command.

The command information list contains a one byte entry for the number of parameters and the modes that a command has associated with it. The low four bits indicate the number of parameters to expect, while the least significant bit of the upper four bits indicates that the command is capable of invoking either mode "01" or mode "10" data transfers. Next, it contains the address of the command

status byte. Then the lower and upper bounds for each of the parameters are listed in sequential order. The actual memory address of the task associated with this command is the last item of the list (see Figure 10).

When a command has been validated and the address of its information list retrieved, the next test of the command checks to insure that the number of parameters indicated in the Cromemco word P/E field is correct. If this test is passed, the command status byte address from the command information list is used to set the most significant bit of the command status byte located in memory. This indicates that this command is now active. Only after passing each of these tests is the command word echoed to the Nova. If any test fails, the command word is echoed with the appropriate error code.

The next test checks the mode of the command. If a command is valid for all modes, the data mode bit will be set in the first byte of the command information list. This byte is the same byte that was used when checking the received parameter count against the expected parameter count. Since commands implementing the data transfer modes ("01" or "10") must always have an extra parameter associated with them, each command will have an extra pair of bounds in the list. These bounds will always appear as the first pair in the information list. Since the routine's Command Information List Pointer was left pointing to the

```
COMTBL: DW      CIL000          ;CIL - TASK 00 - OUTPUT
        DW      CIL00I          ;CIL - TASK 00 - INPUT
        DW      CIL010          ;CIL - TASK 01 - OUTPUT
        DW      CIL01I          ;CIL - TASK 01 - INPUT
;
;       ALL INVALID TASK NUMBERS HAVE A LIST ADDRESS OF ZERO.
;
        DW      0,0,0,0, ... ,0, ... ,0,0,0,0,0,0,0,0,0,0,0
;
;       THE FOLLOWING ARE THE COMMAND INFORMATION LISTS.
;
CIL000: DB      010H            ;LO NIBBLE - PARAMETER COUNT
                                ;HI NIBBLE - DATA FLAG SET
        DW      CMSTAT          ;STATUS LOCATION FOR OUTPUT
                                ;TASK 00
        DW      00000H          ;DATA COUNT LO BOUND
        DW      00400H          ;DATA COUNT HI BOUND
                                ;NO ACTUAL PARAMETERS SINCE
                                ;PARAMETER COUNT IS ZERO
                                ;DATA COUNT INCLUDED SINCE
                                ;DATA FLAG IS SET
        DW      TSK000          ;ADDRESS OF TASK 00
;
CIL00I: DB      10H             ;ZERO PARAMETERS - SET DATA
        DW      CMSTAT+1        ;STATUS LOCATION
        DW      00000H          ;DATA COUNT LO BOUND
        DW      00400H          ;DATA COUNT HI BOUND
        DW      TSK00I          ;TASK ADDRESS
;
CIL010: DB      012H            ;TWO PARAMETERS - SET DATA
        DW      CMSTAT+2        ;STATUS LOCATION
        DW      00000H          ;DATA COUNT LO BOUND
        DW      00400H          ;DATA COUNT HI BOUND
        DW      00070H          ;ONE LESS THAN THE MINIMUM
                                ;SAMPLE RATE SETTING
        DW      0FFFFH          ;MAX. SAMPLE RATE SETTING
        DW      00000H          ;ONE LESS THAN THE LOWEST
                                ;A/D CHANNEL
        DW      00010H          ;HIGHEST A/D CHANNEL
        DW      TSK010          ;TASK ADDRESS
;
CIL01I: DB      012H            ;TWO PARAMETERS - SET DATA
        DW      CMSTAT+3        ;STATUS LOCATION
        DW      00000H          ;DATA COUNT LO BOUND
        DW      00400H          ;DATA COUNT HI BOUND
        DW      00070H          ;ONE LESS THAN THE MINIMUM
                                ;SAMPLE RATE SETTING
        DW      0FFFFH          ;MAX. SAMPLE RATE SETTING
        DW      00000H          ;ONE LESS THAN THE LOWEST
                                ;D/A CHANNEL
        DW      00004H          ;HIGHEST D/A CHANNEL
        DW      TSK01I          ;TASK ADDRESS
```

Figure 10. CHOPS Command Information List Format

low byte of the low bound of the first pair in the list, an error would occur in parameter bounds checking if this pair were used for the non-data modes ("00" or "11") for the same command. Therefore, when the same command is received in a non-data mode, the command information list pointer must be advanced four bytes to effectively skip over the extra parameter bounds. The routine then goes into a loop, receiving and validating parameters and echoing the status of each validation to the Nova.

After all parameters have been collected and placed into the parameter buffer, the task address is extracted from the information list. The mode is once again extracted from the command word and used to generate an offset from the task entry point address. Mode "00" commands enter the task with no offset, while each of the other modes enter two bytes progressively into the task.

Each task must have four jump relative instructions at its beginning which are used to vector the task to the appropriate command service routine (see Figure 11). The task must also provide the command service routine as a proper Z-80 subroutine that executes a return from subroutine when it is completed. If an error occurs during subroutine execution, the CHOPS requires that the subroutine perform a proper return with the carry flag set and the E register loaded with the proper error code. For a non-error condition, the subroutine must return with the carry flag

```
        ENTRY   TSK000,TSK00I
;
;       OUTPUT TASK 00
;
TSK000: JR      OMODE0          ;MODE "00" ENTRY POINT
        JR      OMODE1          ;MODE "01" ENTRY POINT
        JR      OMODE2          ;MODE "10" ENTRY POINT
OMODE3: JP      TCMPLT          ;MODE "11" ENTRY POINT
;
OMODE0: LD      E,ERR6          ;MODE "00" NOT AVAILABLE
                                ;SET "INVALID COMMAND MODE"
        JP      ERROR           ;TAKEN TO HANDLE ERROR
;
OMODE1: LD      DE,DATOUT       ;GET ADDRESS OF MODE "01"
                                ;OUTPUT DATA HANDLER
        JR      GTOJOB          ;TAKEN TO CONTINUE TASK
                                ;SET UP
;
OMODE2: LD      DE,DCHOUT       ;GET ADDRESS OF MODE "10"
                                ;OUTPUT DATA HANDLER
GTOJOB: CALL    SBUFDF          ;SET BUFFER DEFAULTS
        LD      HL,JOB00        ;GET JOB ADDRESS
        JP      DOJOBO          ;LET THE CHOPS DO THE WORK
;
;       INPUT TASK 00
;
TSK00I: JR      OMODE0          ;MODE "00" ENTRY POINT
        JR      IMODE1          ;MODE "01" ENTRY POINT
        JR      IMODE2          ;MODE "10" ENTRY POINT
        JR      OMODE3          ;MODE "11" ENTRY POINT
;
IMODE1: LD      DE,DATINP       ;GET ADDRESS OF MODE "01"
                                ;INPUT DATA HANDLER
        JR      GTIJOB          ;TAKEN TO CONTINUE TASK
                                ;SET UP
;
IMODE2: LD      DE,DCHINP       ;GET ADDRESS OF MODE "10"
                                ;INPUT DATA HANDLER
GTIJOB: CALL    SBUFDF          ;SET BUFFER DEFAULTS
        LD      HL,JOB00        ;GET JOB ADDRESS
        JP      DOJOBI          ;LET THE CHOPS DO THE WORK
;
;       JOB 00 CONSISTS OF A NULL JOB.
;
JOB00:  SCF                     ;CLEAR CARRY FLAG SO
        CCF                     ;CHOPS THINKS ALL IS OK
        RET
        END                     ;END OF TASK 00
```

Figure 11. CHOPS Task Coupling Technique

cleared. The task normally places the address of its service subroutine in the HL register and the address of the CHOPS data transfer routine that is specified by the command mode in the DE register. These addresses are then passed back to CHOPS via an absolute jump to DOJOBO or DOJOBI for output and input tasks respectively. The CHOPS will then handle calling of the service routine and the invoking of the required data handler at the appropriate stage of command execution. If the task requires an optimized data transfer capability, the task can intercede at this point and not return to the CHOPS in the normal manner. When generating such tasks great care should be taken to insure that the channel communication protocol for data transfer is strictly observed. Also, for these tasks, an ultimate return to CHOPS is required by an absolute jump to the task complete (TCMPLT), or error (ERROR), routines. If an error has occurred, the code for the error must be placed in the E register before accomplishing the jump to ERROR.

The data handling routines exist as five distinct routines. Two of these routines allow data to be transferred via programmed I/O. DATINP is for inputting data, while DATOUT is for outputting data. Likewise, two other routines provide data transfer via the DCH capability of the Nova. Again, DCHINP is for input, while DCHOUT handles output. The fifth routine is the link routine which allows the input or output service routines of the tasks to be linked to their respective data handlers. This linking

routine has two entry points. DOJOBI is the entry point for an input task, while DOJOBO is the entry point for an output task. Both these entries expect the address of the data handler to be in the DE register and the address of the job or service routine to be in the HL register. With this information, the linking routine will properly complete the servicing of the command and perform the proper return to the command/parameter collector.

The error handling routine also has several entry points. These entry points are required for handling the many possible error modes. The task completion, TCMPLT, entry point is also embedded into the error handler because its function of deactivating a completed task is very similar. The DE registers are used by the error handler to indicate the bits to be set in the echoed command word. The D register normally contains the error and status bits, while the E register contains the error code. These two registers are "OR"ed onto the least significant byte of the command word to indicate the proper condition immediately before the command is echoed. The error handler also incorporates a test for abort commands. Any time that an improper response is received during a transaction, it is vectored to this routine to test for an abort command. If the abort is detected, the command will be sent to the command/parameter validation and collection routines for servicing.

During their execution, the main routines of CHOPS call numerous subroutines that are common to their functions, but the majority of the interchanges are accomplished through absolute jump instructions. These jumps are necessary because an error condition could occur at any time. If one were to occur in a subroutine that was deeply nested, it could be very difficult to properly adjust the stack before servicing the error. This could lead to stack overflow and ultimately to catastrophic failure of the CHOPS. With the jump technique, each of the modules can stand alone with control being safely passed between them without the concern of causing a misadjusted stack.

The CHOPS has been written to support a possible update to multi-tasking. This is the primary reason that a command status table and a capability for aborting commands were incorporated. For multi-tasking, an interrupt capability would have to be added to CHOPS since it currently does not use any interrupts for its own controlling purposes. The interrupt system of the Cromemco is, however, fully operational and can be employed by individual tasks. When using interrupts, extreme care should be taken to insure that the interrupts are disabled before returning from a command service subroutine. If this is not done, the CHOPS will have no way of regaining control of the system.

IV. Conclusions and Recommendations

The I/O channel, in its final form, meets all the design goals that were set for the joint thesis. The hardware provides a reliable path over which the Nova and Cromemco can communicate. This communication can be accomplished on the Nova's side of the channel with either programmed I/O or the DCH, a major objective of the thesis. A means of transferring speech data onto and off of the Nova's disk was developed early enough to provide a data aquisition capability for other thesis topics which had to be completed months before this one, another major objective. Later, software was developed which allowed the Nova to control the I/O Channel without the interaction of an operator at the Cromemco, which also was a design goal. The CHOPS provides extensive error checking and allows for the addition of new tasks with relative ease. In addition, the individual writing the task does not have to be intimately familiar with the CHOPS or the I/O Channel protocol. The Nova CHANNEL subroutine handles all communication with the CHOPS and relieves the Nova user from having to understand or implement the communication protocol. This feature makes using the channel a simple subroutine call, another design goal. In addition, new capabilities added to the I/O Channel via the Cromemco do not require changes in CHANNEL, again a major design goal.

One self imposed objective, that was not one of the original design goals, was to transfer speech data from the Nova's disk in real-time. This required the data to be transferred from the Nova's disk at a rate equal to or greater than the desired output rate of the D/A converter. Accomplishing this would allow the continuous generation of speech for a duration limited only by the storage capacity of the disk. To test the feasibility of such a capability, the DCH transfer rate had to be determined. Unfortunately, time did not allow for more than obtaining a crude estimate of the DCH transfer time. When a random file was transferred, it took slightly less than four seconds to transfer about four seconds of speech data. With contiguous files the transfer took approximately two seconds for the same amount of speech data. Even though these measurements were quite coarse, they revealed great promise for a real-time transfer capability. Hopefully, further development will provide this capability in the very near future.

One discouraging problem was discovered during the development of the CHOPS. The BUSY and DONE flags which were used as handshake signals for transfers between the Nova and Cromemco seem to malfunction at times. The Cromemco is required to set the DONE flag at the completion of a task, but sometimes it is unable to set this flag. This is rather puzzling since no problems were ever encountered when the DONE flag is used during a DCH transfer, even though the handshake is much faster and

considerably more critical. Several attempts were made to rectify this problem with hardware, but none were successful. The problem seems to originate from the Cromemco using the clock input of the DONE flip-flop to set the flag, while the Nova is affecting this flag via the direct set and direct clear of the flip-flop. It is suggested that a logic analyzer be used to more thoroughly examine the problem. Unfortunately, time constraints again did not permit this trouble shooting to be accomplished. To circumvent this problem the CHOPS was modified to test the DONE flag after each time it attempts to set it. If it is set the CHOPS will continue, but if it is not set the CHOPS will continue to attempt to set the flag until it is finally set. It was discovered that this loop can continue for up to 100 milliseconds, it allows the I/O Channel to function until the problem can be found and eliminated. If the pre-wired BUSY/DONE network should prove to be the problem, it will have to be redesigned and installed in the breadboard section of the GPI.

It should be emphasized that the communications protocol, CHANNEL, and CHOPS are all orientated toward multi-tasking. They were designed to make the ultimate implementation of multi-tasking as easy as possible. Multi-tasking was not implemented at this time since it requires that the interrupt capabilities and their associated handlers be developed for both systems. Considering the

already extensive nature of this thesis, it would not have been possible to complete an additional development of this magnitude in the time available. Before adding multi-tasking, the hardware to allow the Cromemco to load the Nova's B Port with a DCH address should be implemented on the GPI. This capability would allow the I/O Channel to receive an interrupt and extract command/parameter information from a pre-defined area of the Nova's memory. After executing the command the I/O Channel could transfer the completion information into the pre-defined Nova memory area and interrupt the Nova's operation to indicate that the command has completed. Utilization of this technique would speed up the transfer of command information and provide part of the basis for a multi-tasking environment.

The CHOPS software should be placed in ROM. It was designed to reside in ROM to be protected from being over-written. This would also allow the removal of the terminal and disk system currently attached to the Cromemco. The CHOPS would automatically be invoked when the system is powered up making the I/O Channel immediately available to the Nova.

It is suggested that a software development system for the I/O Channel be added to the Speech Lab. The Cromemco was used heavily by other thesis students, making it difficult to develop additional software on the system. A separate development system, configured much like the I/O

Channel itself, would avoid this problem. CHOPS has provisions in it for communicating with such a system via a serial port. This would allow software to be written and debugged in the external system and then downloaded into the Cromemco for running final tests.

When these enhancements have been accomplished, the I/O Channel will provide nearly unlimited capabilities for the Speech Lab user.

Bibliography

1. Bursky, Dave. The S-100 Bus Handbook. Rochelle Park, N.J.: Hayden Book Company, Inc., 1980.

2. Elmquist, Kells A., Howard Fullmer, David B. Gustavson, and George Morrow. "Standard Specification for S-100 Bus Interface Devices," S-100 Micro Systems,1: 20-44 (January/February 1980).

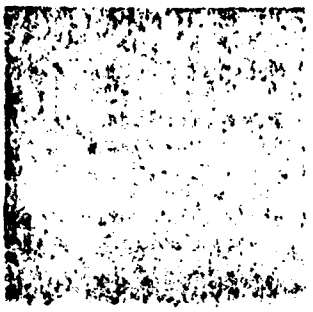
3. Poe, Elmer C. and James C. Goodwin. The S-100 Bus & Other Micro Buses. Indianapolis: Howard W. Sams & Co., Inc., 1979.

4. 014-000629-00. User's Manual, Interface Designer's Reference, NOVA AND ECLIPSE LINE COMPUTERS. Westboro, Mass.: Data General Corporation, December 1978.

Appendix A

IEEE S-100 Bus Standard

MICROSYSTEMS



PROPOSED STANDARD FOR THE S-100 BUS

Preliminary Specification

George Morrow, Thinker Toys
Howard Fullmer, Parasitic Engineering, Inc.

Members, IEEE Computer Society Microprocessor Standards Committee

The computer bus commonly known as the S-100 was first introduced by MITS, Inc., with its Altair kit. This bus has since spread throughout the electronics industry and beyond. Today over a hundred manufacturers make products which claim to be compatible with the S-100 bus even though—until now—no complete specification has been available. The following table, figures, and notes constitute the preliminary draft of a proposed standard for the S-100 bus.

This document is a specification for both timing and signal disciplines. Signal discipline is described using the bus master/bus slave language long associated with Digital Equipment Corporation's PDP-11. This point of view facilitated the development of a simple and highly reliable DMA protocol, the extended addressing capabilities, and the 16-bit wide data path proposals. These extensions to the original Altair bus represent a significant advance to the state of the art of small computers and are a direct result of a continuing dialogue with a large number of interested people who have contributed their thoughts and ideas to the standards committee. The extended address and data proposals are compatible with systems that don't use these features, including most existing systems. Signals which are defined or redefined for the extensions are indicated by an asterisk

The preliminary specification will be presented at the 1978 NCC in June in Anaheim, California. Comments can be made at that time or by writing to George Morrow,

S-100 Bus Signal Definitions (preliminary—subject to revision)

PIN NO.	SIGNAL NAME & TYPE	POLARITY	DESCRIPTION
1	+8 volts (B)[1]		Instantaneous minimum greater than 7 volts, instantaneous maximum less than 25 volts, average maximum less than 11 volts
2	+16 volts (B)		Instantaneous minimum greater than 14 volts, instantaneous maximum less than 35 volts, average maximum less than 20 volts
3	XRDY (S)[1,10]	positive	One of two ready inputs to the current bus master. The bus is ready when both these ready inputs are true
4	VI0 (S)[10]	negative	Vectored interrupt line 0
5	VI1 (S)[10]	negative	Vectored interrupt line 1
6	VI2 (S)[10]	negative	Vectored interrupt line 2
7	VI3 (S)[10]	negative	Vectored interrupt line 3
8	VI4 (S)[10]	negative	Vectored interrupt line 4
9	VI5 (S)[10]	negative	Vectored interrupt line 5
10	VI6 (S)[10]	negative	Vectored interrupt line 6
11	VI7 (S)[10]	negative	Vectored interrupt line 7
12	—	—	Not specified
13	—	—	Not specified
14	—	—	Not specified
15	—	—	Not specified
16	—	—	Not specified
17	—	—	Not specified
18	$\overline{\text{STAT DSB}}$ (M)[1,10]	negative	The control signal to disable the 9* status signals [2]
19	$\overline{\text{C/C DSB}}$ (M)[10]	negative	The control signal to disable the 6 command/control signals [3]
20	UNPROT	—	Not specified
21	SS	—	Not specified
22	$\overline{\text{ADD DSB}}$ (M)[10]	negative	The control signal to disable the 16 address signals [4]

S-100 Bus Signal Definitions (preliminary—subject to revision)

PIN NO	SIGNAL NAME & TYPE	POLARITY	DESCRIPTION
23	$\overline{\text{DO DSB}}$ (M)[10]	negative	The control signal to disable the 8 data output signals [8]
24	ϕ_2 (B)	positive	The master timing signal for the bus
25	ϕ_1	—	Not specified
26	PHLDA (M)	positive	A command/control signal used in conjunction with PHOLD to coordinate bus master transfer operations
27	PWAIT (M)	positive	The acknowledge signal to either of the bus ready signals XRDY, PRDY or to a HLT instruction
28	PINTE	positive	Not specified
29	A5 (M)	positive	Address bit 5
30	A4 (M)	positive	Address bit 4
31	A3 (M)	positive	Address bit 3
32	A15 (M)	positive	Address bit 15 (most significant for non-extended addressing)
33	A12 (M)	positive	Address bit 12
34	A9 (M)	positive	Address bit 9
*35	DO1 (M)/A17 (M)/DATA1 (M/S)[11, 12]	positive	Data out bit 1, extended address bit 17, bidirectional data bit 1
*36	DO0 (M)/A16 (M)/DATA0 (M/S)	positive	Data out bit 0, extended address bit 16, bidirectional data bit 0 (least significant)
37	A10 (M)	positive	Address bit 10
*38	DO4 (M)/A20 (M)/DATA4 (M/S)	positive	Data out bit 4, extended address bit 20, bidirectional data bit 4
*39	DO5 (M)/A21 (M)/DATA5 (M/S)	positive	Data out bit 5, extended address bit 21, bidirectional data bit 5
*40	DO6 (M)/A22 (M)/DATA6 (M/S)	positive	Data out bit 6, extended address bit 22, bidirectional data bit 6
*41	DI2 (M)/DATA10(M/S)[6]	positive	Data in bit 2, bidirectional data bit 10
*42	DI3 (M)/DATA11 (M/S)	positive	Data in bit 3, bidirectional data bit 11
*43	DI7 (M)/DATA15 (M/S)	positive	Data in bit 7, bidirectional data bit 15 (most significant)
44	SM1 (M)	positive	The status signal which indicates that the current cycle is an op code fetch
45	SOUT (M)	positive	The status signal identifying the data transfer bus cycle of an OUT instruction
46	SINP (M)	positive	The status signal identifying the data transfer bus cycle of an IN instruction
47	SMEMR (M)	positive	The status signal identifying bus cycles which transfer data from memory to a bus master which are not interrupt acknowledge instruction fetch cycle(s)
48	SHLTA (M)	positive	The status signal which acknowledges that a HLT instruction has been executed

Thinker Toys, 1201 10th Street, Berkeley, CA 94707 or to Robert G. Stewart, 1658 Belvoir Drive, Los Altos, CA 94022.

The committee is currently considering proposals for DMA and interrupt priority specifications. These will be made public in the near future—perhaps at the NCC meeting.

Bus signal notes (see table)

1. There are three types of signals on the S-100 bus. M stands for bus master. Signals designated by (M) are those which a bus master must generate. The bus master currently controlling the bus has the responsibility for faithfully generating, *all* signals of type M during its control of the bus.

S stands for bus slave. A bus slave need generate only that subset of type S signals which are necessary to communicate with bus masters which have the ability to address the slave.

B stands for bus. Any bus signal which is not of type M or S is by default type B. This is not to say that some bus master is not in fact generating one or more type B signals. Rather a type B signal is one that (a) not all bus masters are required to generate, and (b) not any bus slave is required to generate.

A bus master is, by definition, a bus device which generates at least all of the type M signals. A bus slave is a bus device which generates some subset of type S signals. A bus master may also be a bus slave and vice-versa. Memory devices are almost always bus slaves while DMA devices are usually both a bus master (data transfers) and a bus slave (accepting commands). Central processing units are usually bus masters.

2. The 9* status signals are SMEMR, SINP, SM1, SOUT, SHLTA, SSTACK (not specified), $\overline{\text{SWO}}$, SINTA, and SXTRQ.

3. The 6 command/control signals are PHLDA, PSYNC, PDBIN, PINTE (not specified), $\overline{\text{PWR}}$, and PWAIT.

4. The 16 address signals are A15, A14, A13, A12, A11, A10, A9, A8, A7, A6, A5, A4, A3, A2, A1, and A0.

5. Data output is specified relative to a bus master. By definition, data which is transmitted by a bus

master is always data output and occurs on the DO bus or DATA* bus.

6. Data input is specified relative to a bus master. By definition, data which is received by a bus master is always data input and occurs on the DI bus or DATA* bus.

7. A bus cycle is a collection of bus states (BS_a). A bus cycle always starts with a BS_1 state which is followed by a BS_2 state. After BS_2 comes an indeterminate number of $BS_{..}$ (bus wait) states. A bus cycle may have zero $BS_{..}$ states or it may have an arbitrarily large number of $BS_{..}$ states. BS_3 is the bus state which follows $BS_{..}$ (or BS_2 if there are no $BS_{..}$ states present). BS_3 is followed by zero or more BS_0 (bus idle) states. A BS_3 or BS_0 state terminates a bus cycle.

8. The DO bus is the following set of signals: DO7, DO6, DO5, DO4, DO3, DO2, DO1, and DO0.

9. The DI bus is the following set of signals: DI7, DI6, DI5, DI4, DI3, DI2, DI1, and DI0.

10. These signals should be generated by an open collector bus driver capable of sinking at least 24 mA at no more than 0.5 volts.

*11. During the last half of bus state 1 and the first half of bus state 2, the DO lines are used to furnish extended addressing to slaves that can utilize this information.

*12. For 16-bit masters and slaves, the DI and DO lines gang together to form a 16-bit bidirectional data bus called DATA0-15. The DO lines carry the low-order byte while the DI lines accommodate the high-order byte. The configuration of the DI and DO buses is governed by the signals SXTRQ and SIXTN. When both these signals are low, the DI and DO lines become bidirectional. Otherwise, DI carries the data to the current master while DO carries data to the addressed slave.

Signal characteristics

Bus drivers must sink at least 24 mA at no more than 0.5 volts. Except for open collector drivers, they must source at least 2 mA at no less than 2.4 volts.

Bus receivers must have diode clamps to prevent excessive negative excursions. Any bus signal less than 0.8 volts must be recognized as a logic zero, and any signal more than 2 volts must be interpreted as a logic one.

Receivers are to source no more than 0.8 mA at 0.5 volts and are to sink no more than 80 μA at 2.4 volts. The capacitance load of an input from the bus must not exceed 25 pF.

S-100 Bus Signal Definitions (preliminary—subject to revision)

PIN NO.	SIGNAL NAME & TYPE	POLARITY	DESCRIPTION
49	$\overline{\text{CLOCK}}$ (B)	—	2 MHz, 40-60% duty cycle. Not required to be synchronous with any other bus signals
50	GND		Signal and power ground
51	+8 volts (B)		See comments for pin number 1
52	−16 volts (B)		Instantaneous maximum less than −14 volts, instantaneous minimum greater than −35 volts, average minimum greater than −20 volts
53	$\overline{\text{SSWI}}$	—	Not specified
54	$\overline{\text{EXT CLR}}$	negative	A reset signal to rest bus slaves. When this signal goes low it must stay low for at least 3 bus states
55	—	—	Not specified
56	—	—	Not specified
57	—	—	Not specified
58	—	—	Not specified
*59	$\overline{\text{SXTRQ}}$ (M)	negative	The status signal which requests 16-bit wide slaves to respond by asserting $\overline{\text{SIXTN}}$
60	—	—	Not specified
*61	$\overline{\text{SIXTN}}$ (S)	negative	The signal generated by 16-bit slaves in response to the 16-bit request status signal SXTRQ
62	—	—	Not specified
63	—	—	Not specified
64	—	—	Not specified
65	—	—	Not specified
66	—	—	Not specified
67	$\overline{\text{PHANTOM}}$ (B)	negative	A bus signal which disables normal slave devices and enables phantom slaves—primarily used for bootstrapping systems without hardware front panels
68	MWRITE (B)	positive	MWRITE = $\overline{(\overline{\text{PWR}} \cdot \text{SOUT})}$ This signal must follow PWR by not more than 30 ns
69	$\overline{\text{PS}}$	—	Not specified
70	PROT	—	Not specified
71	RUN	—	Not specified
72	PRDY (S)[10]	positive	See comments for pin number 3
73	$\overline{\text{PINT}}$ (S)[10]	negative	The primary interrupt request bus signal
74	$\overline{\text{PHOLD}}$ (M)[10]	negative	The command/control signal used in conjunction with PHLDA to coordinate bus master transfer operations
75	$\overline{\text{PRESET}}$ (B)[10]	negative	The reset signal to reset bus master devices. When this signal goes low it must stay low for at least 3 bus states

S-100 Bus Signal Definitions (preliminary—subject to revision)

PIN NO	SIGNAL NAME & TYPE	POLARITY	DESCRIPTION
76	PSYNC (M)	positive	The command/control signal identifying BS_1. (See bus state comments.)
77	$\overline{PWR}$ (M)	negative	The command/control signal signifying the presence of valid data on DO bus[4] or DATA* bus [12]
78	PDBIN (M)	positive	The command/control signal that requests data on the DI bus[9] or DATA* bus[12] from the currently addressed slave
79	A0 (M)	positive	Address bit 0 (least significant).
80	A1 (M)	positive	Address bit 1.
81	A2 (M)	positive	Address bit 2
82	A6 (M)	positive	Address bit 6.
83	A7 (M)	positive	Address bit 7.
84	A8 (M)	positive	Address bit 8.
85	A13 (M)	positive	Address bit 13
86	A14 (M)	positive	Address bit 14.
87	A11 (M)	positive	Address bit 11
*88	DO2 (M)/A18 (M)/DATA2 (M/S)	positive	Data out bit 2, extended address bit 18, and bidirectional data bit 2
*89	DO3 (M)/A19 (M)/DATA3 (M/S)	positive	Data out bit 3, extended address bit 19, and bidirectional data bit 3.
*90	DO7 (M)/A23 (M)/DATA7 (M/S)	positive	Data out bit 7 (most significant for 8-bit data), extended address bit 23, and bidirectional data bit 7
*91	DI4 (S)/DATA12 (M/S)	positive	Data in bit 4 and bidirectional data bit 12
*92	DI5 (S)/DATA13 (M/s)	positive	Data in bit 5 and bidirectional data bit 13
*93	DI6 (S)/DATA14 (M/S)	positive	Data in bit 6 and bidirectional data bit 14
*94	DI1 (S)/DATA9 (M/S)	positive	Data in bit 1 and bidirectional data bit 9
*95	DI0 (S)/DATA8 (M/S)	positive	Data in bit 0 (least significant for 8-bit data) and bidirectional data bit 8
96	SINTA (M)	positive	The status signal identifying the instruction fetch cycle(s) that immediately follow an accepted interrupt request presented on PINT
97	$\overline{SWO}$ (M)	negative	The status signal identifying a bus cycle which transfers data from a bus master to a slave
98	SSTACK	—	Not specified
99	$\overline{POC}$ (B)	negative	The power-on clear signal for all bus devices. when this signal goes low, it must stay low for at least 3 bus states
100	GND		Signal and power ground

Bus state comments

BS_1 is the initial bus state of a bus cycle. The address lines, data and status, are in a state of flux during most of BS_1, and PSYNC is active starting with the second half of BS_1.

BS_2 is the second bus state when address data, status, and ready signals become stable.

BS_w states occur as needed to synchronize a bus master with a bus slave which has brought one of the ready lines false.

BS_3 is the data transfer state *when a bus master transfers data* to a slave or vice-versa.

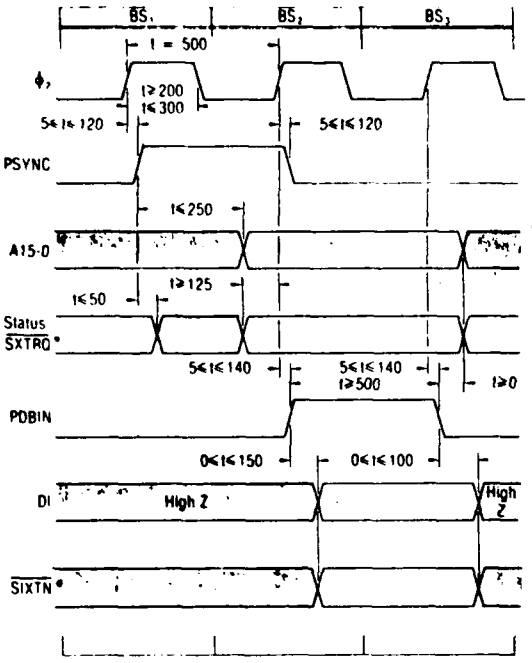
BS_I is a state during which the bus is idle.

Timing notes

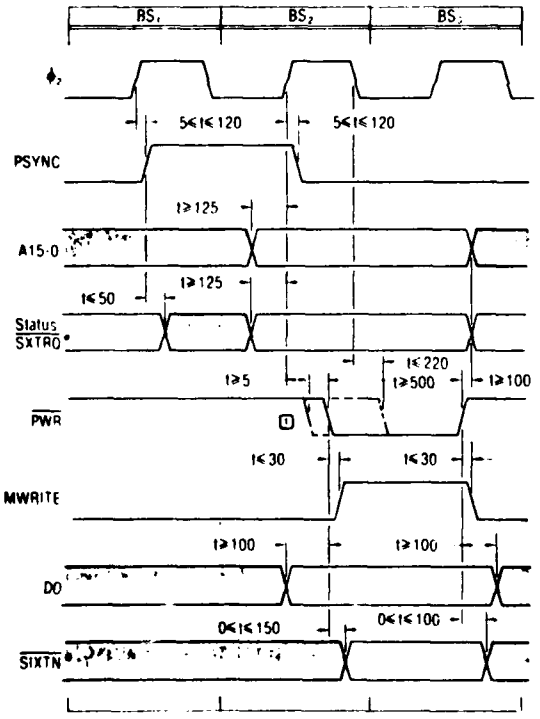
All timing references in the timing diagrams (Figures 1 and 2) are specified at the midpoint of the rising or falling edge of the signal. Rise and fall times are not to exceed 50 ns.

All signals referred to in the timing diagrams are S-100 bus signals with the exceptions in Note 6.

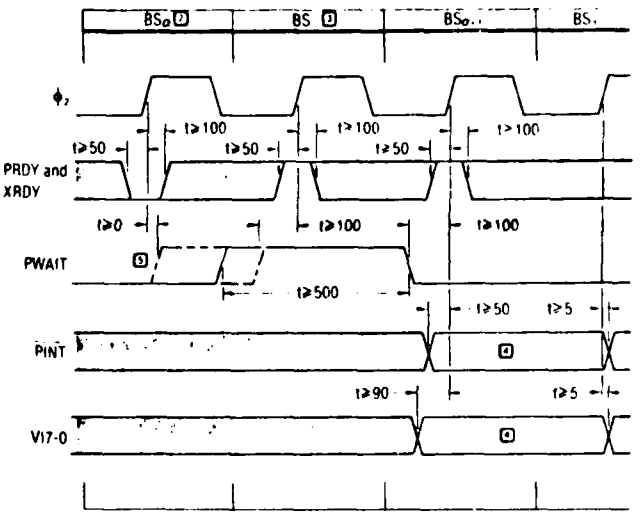
1. The falling edge of $\overline{PWR}$ must occur within the area shown. The rising edge must occur within a similar area of the next bus state.
2. BS_a is either BS_2 or BS_w.
3. Addresses, data output, and status signals must remain stable during BS_w.
4. The interrupt lines must be sampled, by the CPU or interrupt controller, during the stable period shown in the bus state preceding BS_1 of an op-code fetch. This does not imply that the interrupt line must be stable during this period. However, if an interrupt line is asserted true during this period, then it is not defined whether an interrupt will occur before the next instruction is executed. Once an interrupt line is asserted true, it should remain true until the CPU responds. Normally, this response would be an I/O instruction that addresses the interrupting device.
5. The rising edge of PWAIT must occur within the area shown. The falling edge must occur within a similar area following the rising edge of the logical AND of PRDY and XRDY.
6. Signals prefixed by "DMA" refer to internal logic of the new bus master. These signals control the buffers of this bus master which drive the command and control, status, address, and data output bus lines. The timing diagram depicts logic levels which are high when these buffers are disabled.



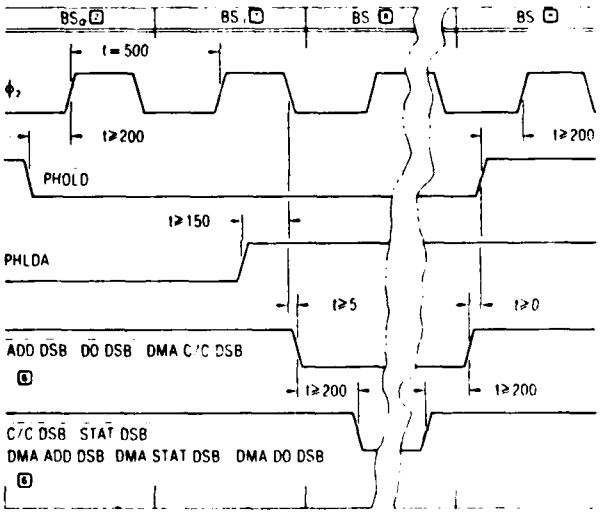
(a) Memory or I/O read.



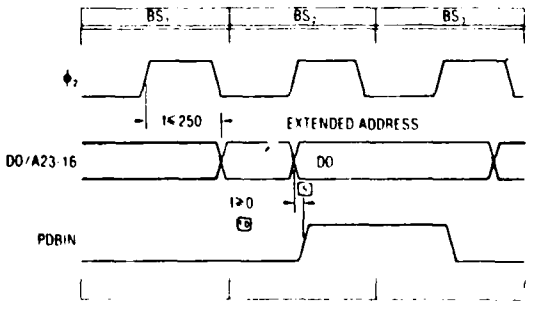
(b) Memory or I/O write.



(c) Interrupt and wait timing.



(d) Bus exchange timing.



*(e) Extended addressing.

Figure 1. S-100 bus timing, in nanoseconds, for (a) memory or I/O read, (b) memory or I/O write, (c) interrupt and wait, (d) bus exchange, and *(e) extended addressing (preliminary—subject to revision).

7. BS_β is either BS_1 or BS_2.

8. BS_γ is either BS_1 or BS_2.

*9. Extended address bits A16 through A23 are multiplexed on the DO lines. They must occur as shown within 250 ns after the rising edge of ϕ_2 during BS_1. During a write, data must appear on these lines 100 ns before $\overline{PWR}$ is active. The $\overline{PWR}$ signal can be asserted as soon as 5 ns after the rising edge of ϕ_2 during BS_2. Therefore, under worst case conditions, the extended addresses are on the bus for 155 ns. Masters can generate $\overline{PWR}$ later as shown. This would allow the extended addresses to be on the bus longer. However, slaves should be designed to correctly respond in the worst case situation.

*10. To avoid conflict on the DO lines during a 16-bit read, similar to the situation described in Note 9 above, the extended addresses must be removed from the DO lines before the rising edge of PDBIN.

Direct memory access requirements

A DMA cycle is a special case of a bus master taking over the bus to execute a memory read or write cycle. A DMA device is required to generate *all* type M (bus master) signals on the bus, and control the bus exchange.

The bus exchange. $\overline{PHOLD}$ is the signal used by one bus master to request that another bus master give up control of the bus. $\overline{PHOLD}$ must not be asserted true unless PHLDA is false.

One bus master (CPU) relinquishes control of the bus to another (DMA) as shown in the bus exchange timing diagram. The DMA device must control the CPU's bus drivers through the use of $\overline{ADD\ DSB}$, $\overline{DO\ DSB}$, $\overline{STAT\ DSB}$, and $\overline{C/C\ DSB}$. It must also control its own bus drivers through the use of signals similar to those shown in Note 6.

The CPU (current master) and the DMA device (new master) must *both* drive the command and control signals for at least 200 ns at two different periods as shown in the bus exchange timing diagram. During these two times, the command and control signals are required to have the following levels: (1) PSYNC = 0; (2) PWAIT = 0; (3) PHLDA = 1; (4) PDBIN = 0; and (5) $\overline{PWR}$ = 1.

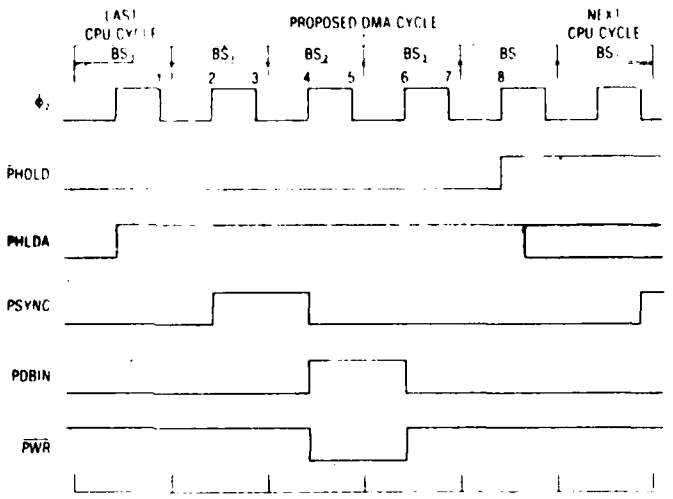


Figure 2. Proposed DMA protocol (preliminary—subject to revision).

The DMA cycle timing sequence which follows is a suggested implementation that meets all the requirements of the generalized bus exchange *timing. The sequence is controlled* by the edges of ϕ_2. At some previous time, $\overline{PHOLD}$ was asserted according to the limitations described in the first paragraph of this section. PHLDA is asserted true by the CPU during BS_1 of the last CPU bus cycle. The bus exchange begins on the falling edge of ϕ_2 while PHLDA is true (labeled 1 on the timing diagram). The DMA bus cycle then proceeds as described in the following section. At edge 8 of ϕ_2, $\overline{PHOLD}$ is driven false by the DMA device and henceforth the CPU is again in control of the bus.

Proposed DMA cycle sequence.

ϕ_2 edge:

1. CPU address and data bus drivers turned off. DMA command and control drivers turned on. The CPU and DMA command and control signals must match the levels described in the previous section.
2. CPU status and command and control drivers turned off. DMA address, data output, and status drivers turned on. PSYNC = 1.
3. No change.
4. PSYNC = 0. PDBIN = 1 if memory read or $\overline{PWR}$ = 0 if memory write.
5. No change
6. PDBIN = 0 and $\overline{PWR}$ = 1.
7. CPU command and control drivers turned on. DMA address and data output drivers turned off.
8. CPU address, data output, and status drivers turned on. DMA status and command and control drivers turned off. $\overline{PHOLD}$ =1.

Multiple data transfers can occur by repeating steps 2 through 6. ■

Acknowledgment

We would like to thank the other members of the Microprocessor Standards Committee for their support and invaluable comments. Special thanks to Robert Stewart who is chairman of the committee. His leadership and organizational skills have been a great aid to all of us.

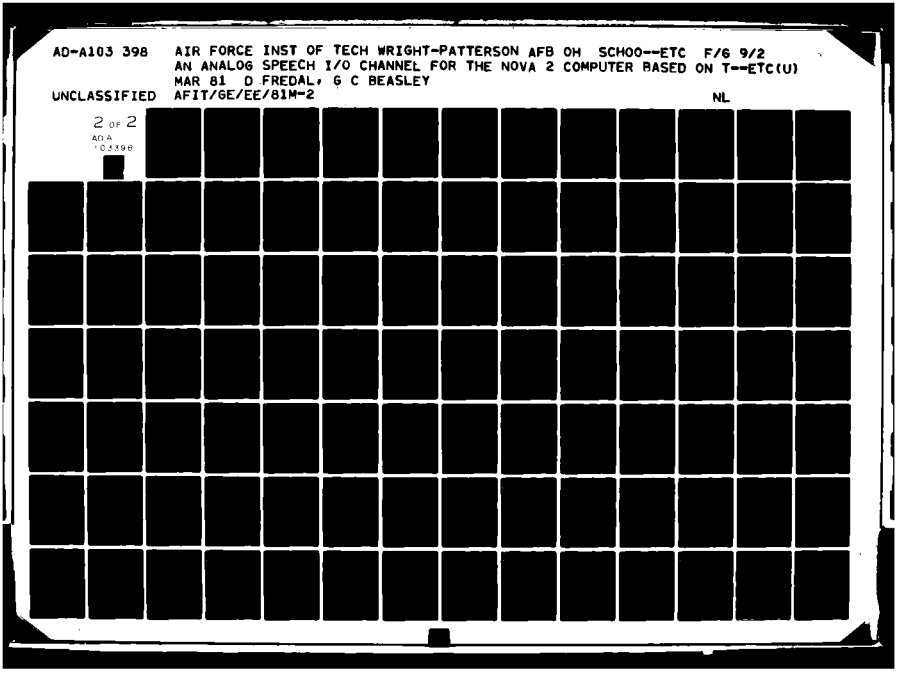
We would also like to extend our thanks to Kells Elmquist and Steve Edleman of Ithaca Audio. Both of these gentlemen flew out to the Second Computer Faire to discuss their ideas about S-100 bus extensions with us. Several of their ideas were truly inspirational. Their intellectual prodding was the force which created the extended addressing and the bidirectional data bus proposals.

Appendix B

Cromemco I/O Port Assignments

Port #	Direction	Function	Card
00	Input	Console status	4FDC
00	Output	Console baud rate	4FDC
01	In/Out	Console data	4FDC
02	Output	Console command	4FDC
03	Input	Console interrupt address	4FDC
03	Output	Console interrupt mask	4FDC
04	In/Out	8-Bit parallel I/O data	4FDC
05	Output	#1 Interval timer	4FDC
06	Output	#2 Interval timer	4FDC
08	Output	#4 Interval timer	4FDC
09	Output	#5 Interval timer	4FDC
30	Input	Disk status	4FDC
30	Output	Disk command	4FDC
31	In/Out	Disk interrupt mask	4FDC
32	In/Out	Disk sector	4FDC
33	In/Out	Disk data	4FDC
34	Input	Disk flags	4FDC
34	Output	Disk control	4FDC
A0	In/Out	DCH low byte data	Crom/Nova
A1	In/Out	DCH high byte data	Crom/Nova
B0	In/Out	DCH address low byte	Crom/Nova
B1	In/Out	DCH address high byte	Crom/Nova
C0	In/Out	Low byte command and data	Crom/Nova
C1	In/Out	High byte command and data	Crom/Nova
C2	Output	DCH transfer request	Crom/Nova
D0	Input	Channel status flags	Crom/Nova
D0	Output	Channel control	Crom/Nova
E0	Input	Master interrupt status #0	CPU Support
E0	Output	Master interrupt control #0	CPU Support
E1	Input	Master interrupt status #1	CPU Support
E1	Output	Master interrupt control #1	CPU Support
E2	Input	Slave interrupt status #0	CPU Support
E2	Output	Slave interrupt control #0	CPU Support
E3	Input	Slave interrupt status #1	CPU Support
E4	Output	Slave interrupt control #1	CPU Support
E4	In/Out	Timing controller data	CPU Support
E5	Input	Timing controller status	CPU Support
E5	Output	Timing controller control	CPU Support
E6	In/Out	Serial data	CPU Support
E7	Input	Serial status	CPU Support
E7	Output	Serial control	CPU Support
E8	In/Out	8-Bit parallel I/O data	CPU Support
E9	Input	8-Bit parallel I/O status	CPU Support
E9	Output	8-Bit parallel I/O control	CPU Support
F0	Output	D/A #4 higher four data bits	D/A
F1	Output	D/A #4 lower eight data bits	D/A
F2	Output	D/A #1 higher four data bits	D/A

AD-A103 398 AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH SCHOO--ETC F/G 9/2
AN ANALOG SPEECH I/O CHANNEL FOR THE NOVA 2 COMPUTER BASED ON T--ETC(U)
MAR 81 D FREDAL, G C BEASLEY
UNCLASSIFIED AFIT/GE/EE/81M-2 NL



Cromemco I/O Port Assignments Cont.

Port #	Direction	Function	Card
F3	Output	D/A #1 lower eight data bits	D/A
F4	Output	D/A #2 higher four data bits	D/A
F5	Output	D/A #2 lower eight data bits	D/A
F6	Output	D/A #3 higher four data bits	D/A
F7	Output	D/A #3 lower eight data bits	D/A
F8	Output	A/D channel select	A/D
F9	Output	A/D control	A/D
F9	Input	A/D status	A/D
FA	Input	A/D lower eight data bits	A/D
FB	Input	A/D higher four data bits	A/D
FF	Output	Video digitizer control	Video

Appendix C

Program Listing - CHOPS

```
;*************************************************************
;
;                        NOVA/CROMEMCO
;
;                  CHANNEL OPERATING SYSTEM
;
;                        C  H  O  P  S
;
;                     MAIN SYSTEM ROUTINES
;                     - - - - - - - - -
;                        VERSION 2.0
;                     - - - - - - - - -
;
;                          WRITTEN BY
;
;                  CAPT GEORGE C. BEASLEY, JR., USAF
;
;                          MARCH 1981
;
;*************************************************************
;
;       THE ONLY EXTERNAL ADDRESS REQUIRED BY THE CHOPS IS
;        THE COMMAND TABLE STARTING ADDRESS.  THE FOLLOWING
;        DECLARATION ALLOWS THIS ADDRESS TO BE PROVIDED
;        DURING LINKING.
;
;       EXT     COMTBL          ;ENTRY POINT FOR THE COM-
;                               ;MAND TABLE AND COMMAND
;                               ;INFORMATION LIST
;
;       THE FOLLOWING LABELS HAVE BEEN MADE GLOBAL FOR USE
;        IN EXTERNAL TASKS.
;
;       THE FOLLOWING ARE THE ENTRY POINTS FOR THE VARIOUS
;        PROTOCOL HANDLING AND LINKAGE ROUTINES.
;
;       ENTRY   TCMPLT,ERROR,DATOUT,DATINP,DCHOUT,DCHINP
;       ENTRY   DOJOBO,DOJOBI,BNDTST,WRNOVA,RDNOVA,TSTDON
;       ENTRY   SBUFDF
;
;       THE FOLLOWING ARE THE ADDRESSES OF THE CHOPS
;        VARIABLE STORAGE LOCATIONS AVAILABLE FOR USE BY
;        EXTERNAL TASKS.
;
;       ENTRY   LOMEM,HIMEM,DATSTR,DATEND,PRMBUF,CMSTAT
;       ENTRY   USERAM
;
```

```
;       THE FOLLOWING ARE THE SYSTEM'S I/O PORT ASSIGNMENTS
;
CONSTS: EQU     000H            ;CONSOLE STATUS PORT
CONDAT: EQU     001H            ;CONSOLE DATA PORT
DCHLO:  EQU     0A0H            ;DCH DATA LO BYTE PORT
DCHHI:  EQU     0A1H            ;DCH DATA HI BYTE PORT
ADDRLO: EQU     0B0H            ;DCH ADDRESS LO BYTE PORT
ADDRHI: EQU     0B1H            ;DCH ADDRESS HI BYTE PORT
DATALO: EQU     0C0H            ;LO BYTE COMMAND AND DATA
DATAHI: EQU     0C1H            ;HI BYTE COMMAND AND DATA
DCHREQ: EQU     0C2H            ;DCH TRANSFER REQUEST PORT
FLAGS:  EQU     0D0H            ;NOVA STATUS FLAGS PORT
DCHDMA: EQU     0D0H            ;DCH AND DMA CONTROL PORT
MSINT0: EQU     0E0H            ;MASTER INTERRUPT
                                ;CONTROLLER - A0 = 0
MSINT1: EQU     0E1H            ;MASTER INTERRUPT
                                ;CONTROLLER - A0 = 1
SLINT0: EQU     0E2H            ;SLAVE INTERRUPT
                                ;CONTROLLER - A0 = 0
SLINT1: EQU     0E3H            ;SLAVE INTERRUPT
                                ;CONTROLLER - A0 = 1
TIMDAT: EQU     0E4H            ;TIMING CONTROLLER DATA
TIMCTL: EQU     0E5H            ;TIMING CONTROLLER CONTROL
                                ;AND STATUS PORT
SERDAT: EQU     0E6H            ;CPU SUPPORT CARD
                                ;SERIAL DATA PORT
SERCTL: EQU     0E7H            ;CPU SUPPORT CARD
                                ;SERIAL CONTROL AND STATUS
PRLDAT: EQU     0E8H            ;CPU SUPPORT CARD
                                ;PARALLEL DATA PORT
PRLCTL: EQU     0E8H            ;CPU SUPPORT CARD
                                ;PARALLEL STATUS PORT
DTABHI: EQU     0F0H            ;#4 D/A DATA HI BYTE PORT
DTABLO: EQU     0F1H            ;#4 D/A DATA LO BYTE PORT
DTACHI: EQU     0F2H            ;#1 D/A DATA HI BYTE PORT
DTACLO: EQU     0F3H            ;#1 D/A DATA LO BYTE PORT
DTADHI: EQU     0F4H            ;#2 D/A DATA HI BYTE PORT
DTADLO: EQU     0F5H            ;#2 D/A DATA LO BYTE PORT
DTAAHI: EQU     0F6H            ;#3 D/A DATA HI BYTE PORT
DTAALO: EQU     0F7H            ;#3 D/A DATA LO BYTE PORT
ATDSEL: EQU     0F8H            ;A/D INPUT SELECT PORT
ATDCVT: EQU     0F9H            ;A/D CONVERSION START PORT
ATDSTS: EQU     0F9H            ;A/D END OF CONVERSION PORT
ATDLO:  EQU     0FAH            ;A/D CONVERTOR LO BYTE PORT
ATDHI:  EQU     0FBH            ;A/D CONVERTOR HI BYTE PORT
;
;
;       THE FOLLOWING LITERAL ASSIGNMENTS SUPPLY THE
;        MNEUMONIC CONSTANTS USED THROUGHOUT CHOPS.
;
MODE0:  EQU     000H            ;MASK FOR SETTING MODE "00"
MODE1:  EQU     010H            ;MASK FOR SETTING MODE "01"
MODE2:  EQU     020H            ;MASK FOR SETTING MODE "10"
MODE3:  EQU     030H            ;MASK FOR SETTING MODE "11"
```

```
ERRBIT: EQU     0C0H            ;MASK FOR SETTING COMMAND
                                ;COMPLETION AND ERROR BITS
CTON:   EQU     0C0H            ;CONTROL CODE FOR CROMEMCO
                                ;TO NOVA DCH
NTOC:   EQU     080H            ;CONTROL CODE FOR NOVA
                                ;TO CROMEMCO DCH
ERR1:   EQU     01H             ;ERROR CODE -
                                ;'NOT A COMMAND'
ERR2:   EQU     02H             ;ERROR CODE -
                                ;'INVALID COMMAND'
ERR3:   EQU     03H             ;ERROR CODE -
                                ;'INVALID PARAMETER COUNT'
ERR4:   EQU     04H             ;ERROR CODE -
                                ;'PARAMETER EXPECTED'
ERR5:   EQU     05H             ;ERROR CODE -
                                ;'PARAMETER OUT OF RANGE'
ERR6:   EQU     06H             ;ERROR CODE -
                                ;'INVALID COMMAND MODE'
ERR7:   EQU     07H             ;ERROR CODE -
                                ;'DATA COMMAND EXPECTED'
ERR8:   EQU     08H             ;ERROR CODE -
                                ;'DATA BUFFER SIZE
                                ;   EXCEEDED'
ERR9:   EQU     09H             ;ERROR CODE -
                                ;'DATA OUT OF RANGE'
ERR10:  EQU     0AH             ;ERROR CODE -
                                ;'REQUESTED DEVICE
                                ;   OFF-LINE'
LF:     EQU     0AH             ;LINE FEED ASCII CODE
CR:     EQU     0DH             ;CARRIAGE RETURN ASCII CODE
ODTCM0  EQU     0FF00H          ;MODE OUTPUT DATA COMMAND
                                ;WITHOUT MODE OR ERROR & S=0
ODTCM1  EQU     0FF40H          ;OUTPUT DATA COMMAND
                                ;WITHOUT MODE OR ERROR & S=1
IDTCM0  EQU     0FF80H          ;INPUT DATA COMMAND
                                ;WITHOUT MODE OR ERROR & S=0
IDTCM1  EQU     0FFC0H          ;INPUT DATA COMMAND
                                ;WITHOUT MODE OR ERROR & S=1
DTCM0   EQU     0FF00H          ;DATA COMMAND WITHOUT
                                ;MODE OR ERROR & S=0
DTCM1   EQU     0FF40H          ;DATA COMMAND WITHOUT
                                ;MODE OR ERROR & S=1
;
;
;       THE FOLLOWING ADDRESS ASSIGNMENTS ARE USED BY CHOPS
;        TO COMMUNICATE WITH EXTERNAL SYSTEM ROUTINES.
;
CDOS:   EQU     00000H          ;CDOS WARM ENTRY POINT
CDSSYS: EQU     00005H          ;CDOS SYSTEM ENTRY POINT
STACK:  EQU     00100H          ;ABSOLUTE TOP OF STACK
;
;
```

```
        ORG     00100H
;
;
;       THIS RAM BASED VERSION HAS THE FOLLOWING ENTRY
;        POINT JUMP VECTORS.
;
BEGIN:  OUT     (040H),A        ;TURN OFF RDOS ROM
        JP      INIT            ;CHOPS COLD START
                                ;  ENTRY POINT
        OUT     (040H),A        ;TURN OFF RDOS ROM
        JP      START           ;CHOPS WARM START
                                ;  ENTRY POINT
;
;
;       REGARDLESS OF WHETHER CHOPS RESIDES IN RAM OR ROM
;        THE FOLLOWING VARIABLES MUST ALWAYS RESIDE IN RAM.
;
CURCOM: DW      00000H          ;CURRENT COMMAND STORAGE
CMTPTR: DW      00000H          ;COMMAND TABLE POINTER
CILPTR: DW      00000H          ;COMMAND INFO LIST POINTER
CSTPTR: DW      00000H          ;COMMAND STATUS POINTER
PRMBPT: DW      00000H          ;PARAMETER BUFFER POINTER
BLKSIZ: DW      00000H          ;DCH BLOCK SIZE STORAGE
DATSTR: DW      00000H          ;ADDRESS OF FIRST DATA WORD
                                ;IN THE DATA BUFFER
DATEND: DW      00000H          ;ADDRESS OF LAST DATA WORD
                                ;IN THE DATA BUFFER
DATTST: DW      00000H          ;ADDRESS OF CURRENT DATA
                                ;BUFFER END
LOMEM:  DW      01000H          ;LOWEST ADDRESS AVAILABLE
                                ;FOR DATA STORAGE
HIMEM:  DW      0BFFFH          ;HIGHEST ADDRESS AVAILABLE
                                ;FOR DATA STORAGE
JOBADR: DW      00000H          ;ADDRESS OF CURRENT JOB
;
;
;     THE FOLLOWING BUFFER IS SET ASIDE FOR THE STORAGE
;       OF FLAGS TO INDICATE THE STATUS OF EACH COMMAND.
;       THESE FLAGS WILL BE REQUIRED BY CHOPS WHEN IT IS
;       FURTHER ENHANCED TO PROVIDE MULTI-TASKING.
;
CMSTAT: DS      00100H          ;COMMAND STATUS BUFFER -
                                ;ROOM FOR 256 COMMANDS
;
;
;       THE FOLLOWING BUFFER IS USED BY CHOPS TO STORE THE
;        PARAMETERS ASSOCIATED WITH EACH COMMAND.
;
PRMBUF: DS      020H            ;PARAMETER BUFFER -
                                ;ROOM FOR 16 WORDS
;
;
```

```
;       THE FOLLOWING RAM SPACE IS SET ASIDE FOR USE BY
;        TASKS.  THIS AREA CAN BE USED BY A TASK FOR
;        RELOCATABLE SERVICE ROUTINES OR GENERAL TASK
;        VARIABLE STORAGE.
;
USERAM: DS      00300H-$        ;THIS MUCH RAM FOR TASK USE
;
;
;    ************************************************************
;    ***** ALL CODE FROM HERE ON CAN BE PLACED IN ROM *****
;    ************************************************************
;
;       THIS IS THE ACTUAL STARTING LOCATION OF THE CHOPS
;
        ORG     00300H          ;CHOPS OBJECT CODE START
;
;       THE FOLLOWING SEQUENCE OF INSTRUCTIONS SETS UP
;        CHOPS ON A COLD START.
;
INIT:   LD      SP,STACK        ;INITIALIZE STACK POINTER
        CALL    SETSER          ;INITIALIZE SERIAL PORT
                                ;IN THIS CASE THE EXTERNAL
                                ;TERMINAL.  ULTIMATELY, FOR
                                ;THE SERIAL LINK TO THE
                                ;DEVELOPMENT SYSTEM.
        LD      HL,COMTBL       ;GET DEFAULT COMMAND
                                ;TABLE ADDRESS
        LD      (CMTPTR),HL     ;SET COMMAND TABLE POINTER
        LD      HL,CMSTAT       ;CLEAR COMMAND STATUS BUFFER
        XOR     A               ;ZERO "A" REGISTER
        LD      B,A             ;ZERO "B" REGISTER
ILOOP:  LD      (HL),A          ;STORE ZERO FOR ALL TASKS
        INC     HL              ;STATUS TO SET TO
        DJNZ    ILOOP           ;INACTIVE STATE
;
;
;       COMMAND/PARAMETER COLLECTION AND VALIDATION ROUTINE
;
;       THE FOLLOWING PORTION OF THE OPERATING SYSTEM IS
;        RESPONSIBLE FOR COLLECTING AND VALIDATING COMMANDS
;        AND PARAMETERS THAT ARE SENT FROM THE NOVA.  THIS
;        SECTION INSURES THAT THE NOVA STRICTLY ADHERES TO
;        THE I/O CHANNEL PROTOCOL FOR COMMAND AND PARAMETER
;        COMMUNICATION.
;
START:  CALL    RDNOVA          ;CHECK FOR NOVA COMMAND
        JR      C,NOVCOM        ;TAKEN TO SERVICE COMMAND
                                ;FROM NOVA
        CALL    RDSERI          ;CHECK FOR ACTIVITY AT
                                ;THE SERIAL PORT
        JR      Z,START         ;TAKEN IF NO COMMAND
                                ;DETECTED AT SERIAL PORT
        OR      A               ;CLEAR AC BEFORE
                                ;ENTERING CDOS
```

```
        JP      CDOS            ;TAKEN TO SERVICE SERIAL
                                ;COMMAND.  NOW MERELY AN
                                ;EXIT TO CDOS.  UTIMATELY
                                ;A DEVELOPMENT STATION
                                ;SERVICE ROUTINE SHOULD
                                ;BE INVOKED AS THE RESULT
                                ;OF A SERIAL COMMAND.
;
NOVCOM: LD      A,B             ;GET COMMAND ID
        AND     OFFH            ;SET CONDITION CODES
        JP      M,ACTCOM        ;TAKEN IF VALID
                                ;COMMAND ID (MSB=1)
        LD      BC,OFFC1H       ;LOAD ECHO COMMAND WITH
                                ;"NOT A COMMAND" ERROR CODE
        JP      ERROR1          ;TAKEN TO HANDLE ERROR
;
ACTCOM: LD      (CURCOM),BC     ;SAVE ALTERED COMMAND WORD
        RLC     C               ;PUT DIRECTION BIT IN CARRY
        RL      B               ;GET DIRECTION AS LSB
                                ;OF COMMAND
        RRC     C               ;RESTORE THE LSB OF THE LO
                                ;BYTE OF THE COMMAND
        LD      HL,(CMTPTR)     ;POINT TO BASE ADDRESS OF
                                ;COMMAND INFO LIST (CIL)
                                ;POINTER TABLE
        LD      E,B             ;USE COMMAND FOR OFFEST
                                ;INTO COMMAND TABLE
        LD      D,00H           ;ZERO HI BYTE OF OFFSET
        SLA     E               ;MULTIPLY LO BYTE BY TWO
        RL      D               ;PICK UP CARRY IN HI BYTE
        ADD     HL,DE           ;POINT TO THIS COMMAND'S
                                ;CIL POINTER
        LD      E,(HL)          ;GET LO BYTE OF CIL ADDRESS
        INC     HL              ;HL NOW POINTS TO ADDRESS
                                ;HI BYTE
        LD      D,(HL)          ;DE POINTS TO CIL
        LD      (CILPTR),DE     ;SAVE CIL POINTER
        LD      A,D             ;GET HI BYTE OF
                                ;ADDRESS OF CIL
        AND     OFFH            ;SET CONDITION CODES
        JR      NZ,VALCOM       ;TAKEN IF COMMAND IS VALID
                                ;HI BYTE IS NON ZERO
        LD      D,ERRBIT        ;GET MASK FOR SETTING ERROR
        LD      E,ERR2          ;GET INVALID COMMAND CODE
        JP      ERROR2          ;TAKEN TO HANDLE ERROR
;
VALCOM: EX      DE,HL           ;HL GETS CIL POINTER
        LD      A,(HL)          ;GET PARAMETER COUNT
                                ;FROM CIL
        AND     OFH             ;MASK TO LOWER FOUR BITS
        LD      B,A             ;SAVE COUNT FROM LIST
        LD      A,C             ;GET LO BYTE OF COMMAND
        AND     OFH             ;MASK TO LOWER FOUR BITS
        CP      B               ;TEST FOR MATCHING P COUNT
```

```
        INC     HL              ;BUMP CIL POINTER
        JR      Z,PCNTOK        ;TAKEN IF P COUNTS ARE O.K.
        LD      E,ERR3          ;GET ERROR CODE FOR
                                ;"INVALID PARAMETER COUNT"
        JP      ERROR           ;TAKEN TO SEND ERROR TO NOVA
;
PCNTOK: LD      E,(HL)          ;GET LO BYTE OF COMMAND
                                ;STATUS LOCATION FROM CIL
        INC     HL              ;BUMP CIL POINTER
        LD      D,(HL)          ;GET HI BYTE OF COMMAND
                                ;STATUS LOCATION FROM CIL
        INC     HL              ;BUMP CIL POINTER
        LD      (CSTPTR),DE     ;SAVE COMMAND STATUS POINTER
        OR      080H            ;SET STATUS BIT TO ACTIVE
        LD      (DE),A          ;SAVE ACTIVE COMMAND STATUS
                                ;WITH PARAMETER COUNT
        LD      BC,PRMBUF       ;GET PARAMETER BUFFER ADDRESS
        LD      (PRMBPT),BC     ;SET BUFFER POINTER TO TOP
                                ;OF BUFFER
        LD      BC,(CURCOM)     ;GET CURRENT COMMAND
        RES     7,C             ;CLEAR ERROR FLAG OF LO BYTE
                                ;OF COMMAND WORD
        CALL    WRNOVA          ;ECHO COMMAND TO NOVA
        LD      A,C             ;GET LO BYTE OF
                                ;CURRENT COMMAND
        AND     030H            ;KEEP ONLY THE COMMAND MODE
        JR      Z,ZMODE         ;TAKEN FOR MODE "00"
        CP      030H            ;CHECK FOR MODE "11"
        JR      NZ,INPRAM       ;TAKEN FOR NON MODE "11"
                                ;TO GET THE EXTRA PARAMETER
ZMODE:  PUSH    HL              ;SAVE THE CURRENT CIL POINTER
        LD      HL,(CILPTR)     ;SET POINTER TO THE TOP OF
                                ;THE CIL FOR THIS COMMAND
        LD      A,(HL)          ;GET THE PARAMETER COUNT BYTE
        POP     HL              ;RESTORE ORIGINAL CIL POINTER
        AND     010H            ;MASK ALL BUT EXTRA
                                ;PARAMETER BIT
        JR      Z,TSTPRM        ;TAKEN FOR COMMANDS NOT
                                ;UTILIZING EXTRA PARAMETER
        INC     HL
        INC     HL
        INC     HL
        INC     HL              ;ADJUST CIL POINTER TO BYPASS
                                ;THE EXTRA PARAMETER BOUNDS
        PUSH    HL              ;SAVE CIL POINTER
        LD      HL,PRMBPT       ;GET PARAMETER BUFFER POINTER
        INC     (HL)            ;ADJUST BUFFER POINTER TO SKIP
        INC     (HL)            ;FIRST PARAMETER STORAGE
        POP     HL              ;RESTORE CIL POINTER
        JR      TSTPRM          ;TAKEN TO CHECK FOR
                                ;PARAMETER COLLECTION
;
PRAMLP: DEC     A               ;DECREASE PARAMETER COUNT
        PUSH    AF              ;SAVE PARAMETER COUNT
```

```
         LD      A,C             ;GET LO BYTE OF COMMAND WORD
         AND     0F0H            ;MASK OUT PARAMETER COUNT
         LD      C,A             ;SET COMMAND LOW BYTE
                                 ;WITH NULL PARAMETER FIELD
         POP     AF              ;RESTORE DECREMENTED
                                 ;PARAMETER COUNT
         OR      C               ;PUT NEW PARAMETER COUNT INTO
         LD      C,A             ;PARAMETER COUNT FIELD
INPRAM:  PUSH    BC              ;SAVE COMMAND TO ECHO TO NOVA
GETPRM:  CALL    RDNOVA          ;GET A PARAMETER FROM NOVA
         JR      NC,GETPRM       ;TAKEN TO WAIT FOR PARAMETER
         LD      A,B             ;GET HI BYTE OF PARAMETER
         OR      A               ;SET CONDITION CODES
         JP      P,CHKPRM        ;TAKEN IF VALID PARAMETER
                                 ;WORD - (MSB IS ZERO)
         LD      E,ERR4          ;GET 'PARAMETER EXPECTED'
                                 ;ERROR CODE
         JR      PRMERR          ;TAKEN TO RESPOND TO
                                 ;ERROR CONDITION
;
CHKPRM:  CALL    BNDTST          ;CHECK PARAMETER AGAINST
                                 ;CIL LO BOUND
         JR      C,PRLOOK        ;TAKEN IF PARAMETER IS
                                 ;GREATER THAN ITS LOW BOUND
PRBERR:  LD      E,ERR5          ;GET ERROR CODE FOR
                                 ;'PARAMETER OUT OF BOUNDS'
PRMERR:  POP     BC              ;RESTORE THE SAVED COMMAND
                                 ;TO ADJUST STACK
         JR      ERROR           ;TAKEN TO RESPOND TO
                                 ;ERROR CONDITION
;
PRLOOK:  INC     HL              ;BUMP CIL POINTER TO
         INC     HL              ;POINT TO HI PARAMETER BOUND
         CALL    BNDTST          ;CHECK PARAMETER AGAINST CIL
                                 ;HI BOUND
         JR      C,PRBERR        ;TAKEN IF PARAMETER IS
                                 ;GREATER THAN ITS HIGH BOUND
PRAMOK:  INC     HL              ;BUMP CIL POINTER TO
         INC     HL              ;POINT TO NEXT CIL ENTRY
         PUSH    HL              ;SAVE CIL POINTER
         LD      HL,(PRMBPT)     ;GET PARAMETER BUFFER POINTER
         LD      (HL),C          ;SAVE LO BYTE OF PARAMETER
         INC     HL              ;BUMP PARAMETER BUFFER POINTER
         LD      (HL),B          ;SAVE HI BYTE OF PARAMETER
         INC     HL              ;BUMP PARAMETER BUFFER POINTER
         LD      (PRMBPT),HL     ;SAVE PARAMETER BUFFER POINTER
         POP     HL              ;RESTORE CIL POINTER
         POP     BC              ;RESTORE COMMAND TO BE ECHOED
         CALL    WRNOVA          ;SEND ECHO COMMAND TO NOVA
TSTPRM:  LD      A,C             ;GET LO BYTE OF CURRENT COMMAND
         AND     0FH             ;KEEP ONLY THE PARAMETER COUNT
         JR      NZ,PRAMLP       ;TAKEN TO COLLECT ANOTHER
                                 ;PARAMETER WORD
NOPRAM:  PUSH    DE              ;SAVE COMMAND STATUS POINTER
```

```
        LD      E,(HL)          ;GET LO BYTE OF TASK ADDRESS
        INC     HL              ;BUMP CIL POINTER
        LD      D,(HL)          ;GET HI BYTE OF TASK ADDRESS
        EX      DE,HL           ;SAVE TASK ADDRESS IN HL
        LD      A,C             ;GET LO BYTE OF
                                ;CURRENT COMMAND
        AND     030H            ;EXTRACT COMMAND MODE
        SRA     A               ;RIGHT JUSTIFY MODE AND
        SRA     A               ;MULTIPLY MODE BY TWO
        SRA     A               ;FOR TASK ENTRY OFFSET
        LD      E,A             ;FORM LO BYTE OF OFFSET
        LD      D,00H           ;ZERO THE OFFSET HI BYTE
        ADD     HL,DE           ;FORM TASK ENTRY POINT IN HL
        POP     DE              ;RESTORE COMMAND STATUS POINTER
        LD      A,(DE)          ;GET CURRENT COMMAND STATUS
        JP      (HL)            ;TAKEN TO PASS CONTROL TO
                                ;THE REQUESTED TASK
;
;
;       TASK COMPLETION/ERROR HANDLING ROUTINE
;
;       THE FOLLOWING PORTION OF THE OPERATING SYSTEM
;        PROVIDES THE CAPABLITY OF COMMUNICATING THE
;        COMPLETION OF A TASK OR ERRORS THAT OCCUR TO THE
;        NOVA.  THE ERROR HANDLING ROUTINES PASS THE NOVA
;        THE ERROR CODES THAT ARE PASSED IN THE 'E' REGISTER.
;        THESE ROUTINES THEN INVOKE THE TASK TERMINATION
;        ROUTINE.  THIS ROUTINE PROVIDES FOR THE ORDERLY
;        TERMINATION OF THE TASK THAT IS CURRENTLY ACTIVE.
;        WHEN THERE IS NO ERROR CONDITION, THE TASK
;        TERMINATION ROUTINE IS INVOKED AT THE END OF THE
;        COMMAND SERVICE.
;
TCMPLT: LD      DE,04000H       ;SET ONLY TASK COMPLETION
                                ;BIT IN ERROR BYTES
        JR      DACTCC          ;TAKEN TO PERFORM PROPER
                                ;COMMAND COMPLETION
;
ERROR:  LD      D,ERRBIT        ;SET ERROR BIT CODE
DACTCC: LD      BC,(CURCOM)     ;GET CURRENT COMMAND
DACTIV: LD      HL,(CSTPTR)     ;GET COMMAND STATUS POINTER
        XOR     A               ;ZERO ACCUMULATOR
        LD      (HL),A          ;SET COMMAND STATUS
                                ;TO INACTIVE
ERROR2: LD      A,C             ;GET LO BYTE OF COMMAND
        AND     030H            ;EXTRACT MODE FIELD ONLY
        OR      D               ;OVERLAY ERROR/COMPLETION
                                ;BITS
        OR      E               ;OVERLAY ERROR CODE FIELD
        LD      C,A             ;REPLACE LO BYTE OF COMMAND
ERROR1: CALL    WRNOVA          ;ECHO COMMAND TO NOVA
        LD      A,098H          ;INSURE THAT DCH IS DISABLED
```

```
        OUT     (DCHDMA),A      ;AND THAT DIRECTION IS SET
                                ;FOR THE CROMEMCO TO INPUT
        JP      START           ;TAKEN TO RESUME SCAN FOR
                                ;NEXT COMMAND
;
DATERR: LD      DE,[ERRBIT+MODE1]*256+ERR8   ;GET MODE "01"
                                ;ERROR BYTES
        JR      CMDERR          ;TAKEN TO SERVICE A DATA
                                ;COMMAND ERROR
;
DCHERR: LD      DE,[ERRBIT+MODE2]*256+ERR8   ;GET MODE "10"
                                ;ERROR BYTES
        JR      CMDERR          ;TAKEN TO SERVICE A DATA
                                ;COMMAND ERROR
;
DCMERR: POP     DE              ;RESTORE ERROR BYTES
        LD      E,ERR7          ;GET 'DATA COMMAND EXPECTED'
                                ;ERROR CODE
CMDERR: LD      BC,DTCM1        ;GET DATA COMMAND WITH S=1
        JR      DACTIV          ;TAKEN TO HANDLE ERROR
                                ;DURING DATA COMMAND
;
DABORT: PUSH    DE              ;SAVE ERROR BYTES
        LD      DE,(CURCOM)     ;GET CURRENT COMMAND
        LD      A,C             ;GET LO BYTE OF POSSIBLE
                                ;RECEIVED COMMAND
        AND     MODE3           ;MASK OUT THE MODE OF
                                ;POSSIBLE COMMAND
        CP      MODE3           ;TEST FOR ABORT COMMAND
        JR      NZ,DCMERR       ;TAKEN WHEN DATA COMMAND
                                ;ERROR DETECTED
        LD      A,E             ;GET LO BYTE OF
                                ;CURRENT COMMAND
        AND     OCFH            ;MASK OFF MODE FIELD
        LD      E,A             ;REPLACE LO BYTE OF
                                ;CURRENT COMMAND
        LD      A,C             ;GET RECEIVED WORD LO BYTE
        AND     OCFH            ;MASK OFF MODE FIELD
        CP      E               ;CHECK FOR MATCH
        JR      NZ,DCMERR       ;TAKEN IF NOT ABORT COMMAND
        LD      A,B             ;GET RECEIVED WORD HI BYTE
        CP      D               ;CHECK AGAINST CURRENT
                                ;COMMAND HI BYTE
        JR      NZ,DCMERR       ;TAKEN IF TASK FIELDS
                                ;DO NOT COMPARE
        POP     DE              ;RESTORE ERROR BYTES
        JP      ACTCOM          ;TAKEN TO SERVICE
                                ;ABORT COMMAND
;
;
;
```

```
;       DATA TRANSFER PROTOCOL HANDLING ROUTINES
;
;       THE FOLLOWING PORTION OF THE OPERATING SYSTEM
;        PROVIDES THE ROUTINES FOR PASSING DATA TO AND
;        FROM THE NOVA.  THESE ROUTINES RELIEVE THE TASKS
;        AND, CONSEQUENTLY, ONE WHO MUST WRITE A TASK OF
;        THE REQUIREMENT OF DETAILED KNOWLEDGE OF THE
;        CHANNEL PROTOCOL.
;
;       TASKS NORMALLY ENTER THIS CODE AT 'DOJOBI' FOR JOBS
;        THAT ARE TO RECEIVE DATA FROM THE NOVA AND AT
;        'DOJOBO' FOR JOBS THAT ARE TO SEND DATA TO THE NOVA.
;        IN EITHER CASE THE ENTRY IS MADE VIA AN ABSOLUTE
;        JUMP TO THE ADDRESS OF THE ENTRY POINT WITH THE 'HL'
;        REGISTER SET TO THE ADDRESS OF THE JOB THAT IS TO
;        BE INVOKED AND THE 'DE' REGISTER SET TO THE ADDRESS
;        OF THE DATA HANDLER THAT IS TO BE USED TO TRANSFER
;        THE DATA.
;
;       FOUR DATA HANDLERS ARE AVAILABLE:
;          'DATINP' - TRANSFERS FROM THE NOVA VIA MODE "01"
;          'DATOUT' - TRANSFERS TO THE NOVA VIA MODE "01"
;          'DCHINP' - TRANSFERS FROM THE NOVA VIA MODE "10"
;          'DCHOUT' - TRANSFERS TO THE NOVA VIA MODE "10"
;
;       TASKS MUST INSURE THAT THE PROPER LINKAGE ADDRESSES
;        ARE GENERATED BEFORE ENTERING THIS CODE.  THE
;        ADDRESSES OF 'DOJOBI', 'DOJOBO', 'DATOUT', 'DATINP',
;        'DCHOUT', AND 'DCHINP' HAVE BEEN DEFINED AS ENTRY
;        ADDRESSES; THEREFORE, TASKS MERELY HAVE TO SPECIFY
;        THESE ADDRESSES AS EXTERNAL TO USE THEM.  THE SYSTEM
;        LINKER WILL THEN RESOLVE THE FINAL ADDRESSES AT LINK
;        TIME.
;
;
DOJOBI: LD      (JOBADR),HL     ;SAVE JOB ADDRESS
        LD      HL,(DATEND)     ;GET ADDRESS OF DATA
                                ;BUFFER END
        LD      (DATTST),HL     ;SET UP DATA TRANSFER
                                ;TEST ADDRESS
        JR      CONTJB          ;TAKEN TO CONTINUE TASK
                                ;SERVICE
;
DOJOBO: PUSH    DE              ;SAVE DATA HANDLER ADDRESS
        CALL    DOJOB           ;INVOKE THE JOB ROUTINE
        POP     DE              ;RESTORE ADDRESS OF DATA
                                ;HANDLER
        JR      C,ERROR         ;TAKEN IF ERROR OCCURRED
                                ;DURING JOB EXECUTION
        LD      HL,(DATEND)     ;GET ADDRESS OF LAST DATA
                                ;WORD IN BUFFER
        LD      (DATTST),HL     ;SET UP DATA TRANSFER
                                ;TEST ADDRESS
```

```
CONTJB: EX      DE,HL           ;PUT DATA HANDLER ADDRESS
                                ;IN HL REGISTER
DOJOB:  JP      (HL)            ;INVOKE DATA HANDLER OR
                                ;CURRENT JOB
;
;
DATOUT: CALL    SETBUF          ;SET DATA WORD COUNT AND
                                ;POINTER TO DATA BUFFER
PUTDAT: PUSH    DE              ;SAVE DATA WORD COUNT
        LD      DE,ODTCM0+MODE1 ;SET UP TO WAIT FOR MODE
                                ;"01" OUTPUT DATA COMMAND
                                ;WITH S=0
        CALL    GETDCM          ;GET COMMAND FROM NOVA
        JR      Z,ODCOK         ;TAKEN IF VALID DATA
                                ;COMMAND RECEIVED
        POP     DE              ;POP STACK FOR ERROR EXIT
        LD      D,ERRBIT+MODE1  ;SET UP ERROR BYTE
        JR      DABORT          ;TAKEN TO TEST FOR
                                ;ABORT COMMAND
;
ODCOK:  LD      BC,DTCM0+MODE1  ;SET UP TO ECHO MODE "01"
                                ;DATA COMMAND WITH S=0
        POP     DE              ;GET WORD COUNT
        LD      A,D             ;GET WORD COUNT HI BYTE
        OR      A               ;SET CONDITION CODES
        JR      NZ,PUTNXT       ;TAKEN IF HI BYTE NOT ZERO
        LD      A,E             ;GET WORD COUNT LO BYTE
        CP      01H             ;CHECK FOR LAST WORD
        JR      NZ,PUTNXT       ;TAKEN IF NOT LAST WORD
        SET     6,C             ;SET COMMAND COMPLETION BIT
PUTNXT: CALL    WRNOVA          ;ECHO COMMAND TO NOVA
        CALL    TSTDON          ;WAIT FOR CLEARED DONE FLAG
        CALL    RDNOVA          ;CHECK FOR COMMAND FROM NOVA
        JR      NC,PUTDT        ;TAKEN IF NO NEW COMMAND
        LD      D,ERRBIT+MODE1  ;SET UP ERROR BYTE
        JR      DABORT          ;TAKEN TO TEST FOR
                                ;ABORT COMMAND
;
PUTDT:  CALL    OUTDAT          ;SEND DATA WORD TO NOVA
        CALL    DBETST          ;TEST FOR DATA BUFFER END
        JP      C,DATERR        ;TAKEN IF DATA BUFFER HAS
                                ;OVERFLOWED
        DEC     DE              ;DECREASE WORD COUNT
        LD      A,E             ;GET LOW BYTE OF WORD COUNT
        OR      A               ;SET CONDITION CODES
        JR      NZ,PUTDAT       ;TAKEN IF MORE DATA TO SEND
        CP      D               ;TEST FOR ZERO DATA COUNT
        JR      NZ,PUTDAT       ;TAKEN TO SEND MORE DATA
        JP      TCMPLT          ;TAKEN TO WRAP UP COMMAND
;
;
DCHOUT: LD      DE,ODTCM0+MODE2 ;SET UP TO WAIT FOR MODE
                                ;"10" OUTPUT DATA COMMAND
        CALL    GETDCM          ;GET NOVA DATA COMMAND
```

```
        JR      NZ,ODCHEX       ;TAKEN IF DATA COMMAND
                                ;NOT RECEIVED
        LD      BC,DTCM0+MODE2  ;SET UP TO ECHO MODE "10"
                                ;DATA COMMAND WITH S=0
        SET     6,E             ;SET COMMAND COMPLETION
                                ;BIT OF COMMAND LO BYTE
        CALL    CMDTST          ;ECHO DATA COMMAND
        JR      NC,ODCHGO       ;TAKEN IF NOVA HAS NOT
                                ;SENT A NEW COMMAND
        JR      ODCHEX          ;TAKEN TO EXIT DCH TRANSFER
                                ;AND TEST FOR ABORT COMMAND
;
ODCHGO: LD      A,CTON          ;GET CONTROL CODE FOR
                                ;CROMEMCO TO NOVA
        CALL    DCHSET          ;SET DCH CONTROL BYTE

OTBCNT: LD      DE,(BLKSIZ)     ;GET DCH BLOCK SIZE
                                ;NUMBER OF 512 BYTE NOVA
                                ;DISK BLOCKS
        LD      C,DCHLO         ;GET PORT ADDRESS OF DCH
                                ;LO BYTE
OTDCNT: LD      B,00H           ;ZERO DCH WORD COUNTER
        CALL    DBETST          ;CHECK FOR DATA BUFFER END
        JP      C,DCHERR        ;TAKEN IF PASSED END OF
                                ;DATA BUFFER
ODCHLP: CALL    OUTDCH          ;SEND A WORD VIA DCH
        JP      NZ,ODCHLP       ;CONTINUE FOR 256 WORDS
        DEC     E               ;DECREASE DCH BLOCK COUNT
        JP      NZ,OTDCNT       ;TAKEN TO GET NEXT BLOCK
        LD      DE,ODTCM1+MODE2 ;SET TEST FOR MODE "10" DATA
                                ;COMMAND WITH S=0
        LD      BC,DTCM0+MODE2  ;SET MODE "10" DATA COMMAND
        CALL    CMDTST          ;ECHO DATA COMMAND TO NOVA
        JP      NC,OTBCNT       ;TAKEN IF NEW COMMAND FROM
                                ;NOVA NOT DETECTED
ODCHEX: LD      D,ERRBIT+MODE2  ;SET ERROR BYTE
        JP      NZ,DABORT       ;TAKEN TO TEST FOR ABORT
        LD      BC,DTCM1+MODE2  ;SET MODE "10" DATA
                                ;COMPLETE COMMAND
        CALL    WRNOVA          ;ECHO DATA COMPLETE COMMAND
        JP      TCMPLT          ;TAKEN TO WRAP UP COMMAND
;
;
DATINP: CALL    SETBUF          ;SET DATA WORD COUNT AND
                                ;POINTER TO DATA BUFFER
GETDAT: PUSH    DE              ;SAVE DATA WORD COUNT
        LD      DE,IDTCM0+MODE1 ;SET UP TO WAIT FOR MODE
                                ;"01" DATA COMMAND WITH S=0
        CALL    GETDCM          ;GET COMMAND FROM NOVA
        JR      Z,IDCOK         ;TAKEN IF VALID DATA
                                ;COMMAND RECEIVED
        POP     DE              ;POP STACK FOR ERROR EXIT
        LD      D,ERRBIT+MODE1  ;SET UP ERROR BYTE
```

```
          JP        DABORT              ;TAKEN TO CHECK FOR
                                        ;ABORT COMMAND
;
IDCOK:    LD        BC,DTCM0+MODE1      ;SET UP TO ECHO MODE "01"
                                        ;DATA COMMAND WITH S=0
          CALL      WRNOVA              ;ECHO COMMAND TO NOVA
          CALL      INPDAT              ;GET DATA WORD FROM NOVA
          CALL      DBETST              ;TEST FOR BUFFER END
          JP        C,DATERR            ;TAKEN IF PASSED END OF
                                        ;DATA BUFFER
          LD        BC,DTCM0+MODE1      ;SET UP TO ECHO MODE "01"
                                        ;DATA COMMAND WITH S=0
          POP       DE                  ;GET WORD COUNT
          LD        A,D                 ;GET WORD COUNT HI BYTE
          OR        A                   ;SET CONDITION CODES
          JR        NZ,GETNXT           ;TAKEN IF HI BYTE NOT ZERO
          LD        A,E                 ;GET WORD COUNT LO BYTE
          CP        01H                 ;TEST FOR LAST DATA WORD
          JR        NZ,GETNXT           ;TAKEN IF NOT LAST WORD
          SET       6,C                 ;SET COMMAND COMPLETION BIT
                                        ;IN LO BYTE OF ECHO COMMAND
GETNXT:   CALL      WRNOVA              ;ECHO DATA COMMAND TO NOVA
          DEC       DE                  ;DECREASE WORD COUNT
          LD        A,E                 ;GET LO BYTE OF WORD COUNT
          OR        A                   ;SET CONDITION CODES
          JR        NZ,GETDAT           ;TAKEN IF MORE INPUT DATA
          CP        D                   ;TEST FOR ZERO DATA COUNT
          JR        NZ,GETDAT           ;TAKEN TO GET MORE DATA
          JR        DOIJOB              ;TAKEN TO INVOKE INPUT JOB
;
;
DCHINP:   LD        DE,IDTCM0+MODE2     ;SET UP TO WAIT FOR MODE
                                        ;"10" OUTPUT DATA COMMAND
          CALL      GETDCM              ;GET NOVA DATA COMMAND
          JP        NZ,IDCHEX           ;TAKEN IF DATA COMMAND
                                        ;NOT RECEIVED
          SET       6,E                 ;SET COMMAND COMPLETION BIT
          LD        BC,DTCM0+MODE2      ;SET UP TO ECHO MODE "10"
                                        ;DATA COMMAND WITH S=0
          CALL      CMDTST              ;ECHO DATA COMMAND
          JR        NC,IDCHGO           ;TAKEN IF NOVA HAS NOT
                                        ;SENT A NEW COMMAND
          JR        IDCHEX              ;TAKEN TO EXIT DCH TRANSFER
                                        ;AND TEST FOR ABORT COMMAND
;
IDCHGO:   LD        A,NTOC              ;GET CONTROL CODE FOR
                                        ;NOVA TO CROMEMCO
          CALL      DCHSET              ;SET DCH CONTROL BYTE
INBCNT:   LD        DE,(BLKSIZ)         ;GET DCH BLOCK SIZE
          LD        C,DCHLO             ;SET PORT ADDRESS FOR DCH
                                        ;LO BYTE
INDCNT:   LD        B,00H               ;SET WORD COUNTER TO ZERO
          CALL      DBETST              ;CHECK FOR DATA BUFFER END
```

```
         JP       C,DCHERR          ;TAKEN IF PASSED END OF
                                    ;DATA BUFFER
IDCHLP:  CALL     INPDCH            ;INPUT VIA DATA CHANNEL
         JP       NZ,IDCHLP         ;TAKEN TO CONTINUE
                                    ;RECEIVING DATA BLOCK
         DEC      E                 ;DECREMENT DCH BLOCK COUNT
         JP       NZ,INDCNT         ;TAKEN TO RECEIVE NEXT
                                    ;PORTION OF DCH BLOCK
         LD       DE,IDTCM1+MODE2   ;GET EXPECTED DATA COMMAND
         LD       BC,DTCM0+MODE2    ;GET ECHO DATA COMMAND
         CALL     CMDTST            ;ECHO DATA COMMAND AND CHECK
                                    ;FOR COMMAND FROM NOVA
         JP       NC,INBCNT         ;TAKEN IF NO NEW COMMAND
                                    ;FROM NOVA
IDCHEX:  LD       D,ERRBIT+MODE2    ;SET ERROR BYTE
         JP       NZ,DABORT         ;TAKEN TO TEST FOR ABORT
         LD       BC,DTCM1+MODE2    ;SET MODE "10" DATA
                                    ;COMPLETE COMMAND
         CALL     WRNOVA            ;ECHO DATA COMPLETE COMMAND
DOIJOB:  CALL     TSTDON            ;WAIT FOR DONE TO CLEAR
         CALL     RDNOVA            ;CHECK FOR NEW COMMAND
                                    ;FROM NOVA
         LD       D,ERRBIT          ;SET ERROR BYTE IN CASE OF
                                    ;NEW COMMAND
         JP       C,DABORT          ;TAKEN IF NEW COMMAND FROM
                                    ;NOVA RECEIVED
         LD       (DATEND),HL       ;SAVE DATA BUFFER POINTER
         LD       HL,(JOBADR)       ;GET ADDRESS OF INPUT JOB
         CALL     DOJOB             ;INVOKE THE COMMANDED JOB
         JP       C,ERROR           ;TAKEN IF ERROR DURING
                                    ;JOB EXECUTION
         JP       TCMPLT            ;TAKEN TO WRAP UP COMMAND
;
;
;
;        CHOPS ---- SUBROUTINES
;
;        THE FOLLOWING SUBROUTINES ARE CALLED FROM VARIOUS
;         PLACES WITHIN THE OPERATING SYSTEM.  SEVERAL OF
;         THESE ROUTINES HAVE BEEN MADE GLOBAL AND THEREFORE,
;         CAN BE USED BY TASKS.  USING SOME OF THESE ROUTINES
;         SHOULD SIMPLIFY THE WRITING OF ADDITIONAL TASKS FOR
;         THE CHOPS.
;
;        CONSULT ROUTINE NAMES IN THE ENTRY STATEMENTS AT
;         THE BEGINNING OF THIS LISTING FOR THE ROUTINES
;         THAT ARE GLOBAL.
;
;
;        THE FOLLOWING ROUTINE WAS INTENDED TO BE RESPONSIBLE
;         FOR SETTING UP THE HARDWARE COMPOSING THE SERIAL
;         LINK TO THE DEVELOPMENT STATION.  UNFORTUNATELY,
;         THE DEVELOPMENT STATION WAS NOT AVAILABLE AT THE
```

```
;         TIME THAT THIS VERSION OF THE CHOPS WAS PUBLISHED.
;         HOPEFULLY, A FUTURE VERSION WILL INCORPORATE THIS
;         CAPABILITY.
;
;        THIS ROUTINE CURRENTLY PRINTS THE 'PROMPT' MESSAGE
;         TO THE SYSTEM CONSOLE CONNECTED TO THE SYSTEM
;         DURING CHOPS DEVELOPMENT.
;
SETSER: LD      DE,PROMPT       ;GET ADDRESS OF PROMPT
        LD      C,09H           ;CDOS PRINT BUFFERED LINE
        JP      CDSSYS          ;CDOS SYSTEM ENTRY POINT
;
;
PROMPT: DB      CR,LF,LF,LF,LF,LF,LF,LF,LF,LF,LF
        DB      '          C  H  O  P  S',CR,LF
        DB      '             IS   UP',CR,LF,LF,LF
        DB      'DEPRESSING ANY KEY WILL EXIT',CR,LF,LF
        DB      'TO RESTART -',CR,LF
        DB      '  IF DISK DRIVE IS CONNECTED',CR,LF
        DB      '  TYPE "CHOPS"/RETURN',CR,LF
        DB      '  OTHERWISE - PRESS RESET/RETURN',CR,LF
        DB      '  TYPE "G100"/RETURN',CR,LF,LF,LF,LF,LF,'$'
;
;
;        THIS ROUTINE IS RESPONSIBLE FOR TESTING FOR AND
;         READING DATA ON THE SERIAL DEVELOPMENT STATION
;         LINK.  AS EXPLAINED FOR THE ROUTINE ABOVE THE
;         DEVELOPMENT STATION WAS NOT AVAILABLE FOR THIS
;         VERSION.
;
;        CURRENTLY THIS ROUTINE SIMPLY RETURNS THE ASCII
;         VALUE OF ANY KEY THAT HAS BEEN DEPRESSED ON THE
;         SYSTEM CONSOLE KEYBOARD.  A VALUE OF ZERO IS
;         RETURNED IF A KEY HAS NOT BEEN PRESSED.
;
RDSERI: LD      C,0BH           ;CDOS TEST CONSOLE READY
        CALL    CDSSYS          ;CDOS SYSTEM ENTRY POINT
        OR      A               ;SET CONDITION CODES
        RET
;
;
;        THIS ROUTINE SAVES THE JOB ADDRESS PASSED IN THE HL
;         REGISTER AND THEN FALLS THROUGH TO THE READ NOVA
;         ROUTINE.
;
SVJADR: LD      (JOBADR),HL     ;SAVE JOB ADDRESS
;
;
;        THIS ROUTINE READS THE NOVA STATUS FLAGS CHECKING
;         FOR THE NOVA BUSY FLAG TO BE SET.  IF THE BUSY
;         FLAG IS SET THE ROUTINE RETURNS WITH THE CARRY
;         SET AND THE WORD RECEIVED VIA THE NOVA'S C-PORT
;         IN THE BC REGISTER.
;
```

```
RDNOVA: IN      A,(FLAGS)       ;GET NOVA STATUS FLAGS
        RRA                     ;NOVA BUSY FLAG TO CARRY
        RET     NC              ;RETURN IF BUSY NOT SET
;
        IN      A,(DATAHI)      ;GET HI BYTE OF NOVA WORD
        LD      B,A             ;MOVE TO COMMAND REGISTER
        IN      A,(DATALO)      ;GET LO BYTE OF NOVA WORD
                                ;DONE FLAG WILL BE SET
        LD      C,A             ;MOVE TO COMMAND REGISTER
        RET
;
;
;       THIS ROUTINE CONTINUOUSLY READS THE NOVA STATUS
;        FLAGS UNTIL THE NOVA DONE FLAG IS CLEARED.
;
TSTDON: IN      A,(FLAGS)       ;GET NOVA STATUS FLAGS
        RLA                     ;NOVA DONE FLAG TO CARRY
        JR      C,TSTDON        ;TAKEN IF DONE FLAG SET
        RET
;
;
;       THIS ROUTINE WAITS FOR THE NOVA DONE FLAG TO CLEAR
;        THEN SENDS THE INFORMATION PASSED IN THE BC REGISTER.
;
;  ************************************************************
;  *    BECAUSE OF AN UNDETERMINED PROBLEM IN THE I/O         *
;  *     CHANNEL THIS ROUTINE HAD TO BE MODIFIED TO ENSURE    *
;  *     THAT THE DONE FLAG ACTUALLY GETS SET.  THE CODE      *
;  *     BEGINNING AT THE LABEL 'PATCH' WAS ADDED TO CAUSE    *
;  *     THE CROMEMCO TO LOOP UNTIL IT SENSES A SET DONE      *
;  *     FLAG.  EACH TIME THROUGH THE LOOP A MESSAGE          *
;  *     INDICATING THAT A CLEARED DONE FLAG WAS DETECTED     *
;  *     IS PRINTED TO THE SYSTEM CONSOLE.  THIS PROBLEM      *
;  *     MUST BE CORRECTED AND THIS PATCH REMOVED BEFORE      *
;  *     ATTEMPTING TO PLACE THE CHOPS IN ROM OR REMOVING     *
;  *     THE SYSTEM CONSOLE.                                  *
;  ************************************************************
;
WRNOVA: CALL    TSTDON          ;WAIT FOR DONE TO CLEAR
WRNOV1: LD      A,B             ;GET HI BYTE
        OUT     (DATAHI),A      ;SEND HI BYTE TO NOVA
        LD      A,C             ;GET LO BYTE
        OUT     (DATALO),A      ;SEND LO BYTE TO NOVA
                                ;DONE FLAG IS SET
PATCH:  IN      A,(FLAGS)       ;GET NOVA FLAGS
        RLA                     ;NOVA DONE FLAG TO CARRY
        RET     C               ;RETURN IF DONE FLAG SET
;
        PUSH    DE              ;SAVE DE CONTENTS
        PUSH    BC              ;SAVE BC CONTENTS
        LD      DE,MSG          ;GET ADDRESS OF MESSAGE
        LD      C,09H           ;CDOS PRINT BUFFERED LINE
        CALL    CDSSYS          ;CDOS SYSTEM CALL
        POP     BC              ;RESTORE BC
```

```
        POP     DE              ;RESTORE DE
        JR      WRNOV1          ;TAKEN TO ENSURE DONE FLAG
                                ;IS SET
;
MSG:    DB      'RE-ATTEMPTING TO SET DONE FLAG',CR,LF,'$'
        RET
;
;
;       THIS ROUTINE TESTS THE ABSOLUTE VALUE OF THE WORD
;        PASSED IN THE BC REGISTER WITH RESPECT TO THE WORD
;        STORED IN THE MEMORY LOCATION ADDRESSED BY THE HL
;        REGISTER.  THE CARRY CODE IS SET IF THE WORD IN
;        THE BC REGISTER IS GREATER THAN THE WORD IN MEMORY.
;        IF THE TWO WORDS ARE EQUAL THE ZERO CONDITION CODE
;        WILL BE SET AND THE CARRY CODE WILL BE CLEARED.
;        THE WORD STORED IN MEMORY MUST BE STORED IN THE
;        CONVENTIONAL LO BYTE/HI BYTE FORM (WITH THE LO
;        BYTE IN THE LOWER ADDRESS).
;
BNDTST: INC     HL              ;SET MEMORY POINTER TO THE
        LD      A,(HL)          ;HI BYTE;GET THE BOUND HI BYTE
        DEC     HL              ;POINT TO LO BYTE OF BOUND
        CP      B               ;CHECK DATA HI BYTE AGAINST
                                ;STORED BOUND
        RET     NZ              ;RETURN WITH CARRY SET IF
                                ;NUMBER IN BC REGISTER IS
                                ;LARGER THAN THE BOUND
;
        LD      A,(HL)          ;GET THE BOUND LO BYTE
        CP      C               ;CHECK DATA LO BYTE AGAINST
                                ;STORED BOUND
        RET                     ;RETURN WITH CARRY SET IF
                                ;NUMBER IN BC REGISTER IS
                                ;LARGER THAN THE BOUND
;
;
;       THE FOLLOWING ROUTINE SETS UP THE DATA BUFFER AND
;        PARAMETER BUFFER POINTERS.
;
SETBUF: LD      HL,(DATSTR)     ;GET STARTING ADDRESS OF
                                ;THE DATA BUFFER
        LD      DE,(PRMBUF)     ;GET THE ADDRESS OF THE
                                ;FIRST PARAMETER
        RET
;
;
;       THE FOLLOWING ROUTINE INITIALIZES THE DATA BUFFER
;        POINTER STORAGE LOCATIONS TO INDICATE THAT THE
;        ENTIRE REMAINING MEMORY IS AVAILABLE FOR DATA
;        STORAGE.
;
SBUFDF: LD      HL,(LOMEM)      ;GET ABSOLUTE LOWEST ADDRESS
                                ;AVAILABLE IN RAM SPACE
```

```
        LD      (DATSTR),HL     ;SAVE AS THE DATA BUFFER
                                ;STARTING ADDRESS
        LD      HL,(HIMEM)      ;GET ABSOLUTE HIGHEST ADDRESS
                                ;AVAILABLE IN RAM SPACE
        LD      (DATEND),HL     ;SAVE AS THE DATA BUFFER
                                ;ENDING ADDRESS
        RET
;
;
;       THE FOLLOWING ROUTINE SETS THE DCH/DMA CONTROL
;        REGISTER TO THE BYTE PASSED IN THE ACCUMULATOR.
;        IT ALSO INITIALIZES THE DATA AND PARAMETER
;        BUFFER POINTERS AND SAVES THE DATA CHANNEL
;        BLOCK SIZE THAT IS PASSED IN THE DE REGISTER.
;
DCHSET: OUT     (DCHDMA),A      ;SET DCH/DMA CONTROL REG.
        CALL    SETBUF          ;SET UP DATA BUFFER AND
                                ;PARAMETER BUFFER START
        LD      (BLKSIZ),DE     ;SAVE DCH BLOCK SIZE
        RET
;
;
;       THE FOLLOWING ROUTINE CONTINUOUSLY READS THE NOVA
;        FLAGS AND DOES NOT RETURN UNTIL THE DATA CHANNEL
;        READY FLAG IS SET.
;
DCHRDY: IN      A,(FLAGS)       ;GET NOVA FLAGS
        RLA                     ;MOVE DCH READY FLAG INTO
        RLA                     ;THE CARRY
        JR      NC,DCHRDY       ;TAKEN UNTIL DCH READY
        RET
;
;
;       THE FOLLOWING ROUTINE TRANSMITS THE ECHO COMMAND
;        THAT IS PASSED IN THE 'BC' REGISTER.  IT THEN
;        WAITS FOR THE NOVA TO CLEAR THE DONE FLAG WHICH
;        ACKNOWLEDGES RECEIPT OF THE COMMAND ECHO.  THE
;        ROUTINE THEN ATTEMPTS TO READ A NEW COMMAND
;        FROM THE NOVA.  IF IT DOES NOT FIND A NEW COMMAND
;        THE ROUTINE RETURNS WITH THE CARRY CODE CLEARED.
;        IF A NEW COMMAND IS PRESENT THE ROUTINE WILL
;        CHECK THE RECEIVED COMMAND AGAINST THE EXPECTED
;        COMMAND CONTAINED IN THE 'DE' REGISTER.  IF A
;        MATCH IS FOUND THE ZERO CONDITION CODE IS SET;
;        OTHERWISE, THE ROUTINE RETURNS WITH ONLY THE
;        CARRY CODE SET.  THIS ROUTINE ALSO IS USED TO
;        WAIT FOR AND VALIDATE EACH DATA COMMAND AS IT IS
;        RECEIVED.  THE ENTRY POINT FOR THE LATTER IS AT
;        'GETDCM'.  AFTER RECEIVING A DATA COMMAND THE
;        ROUTINE FUNCTIONS AS FOR ANY OTHER RECEPTION.
;
CMDATA: LD      BC,DTCM1+MODE2  ;SET UP MODE "01" DATA
                                ;COMMAND WITH COMPLETION SET
CMDTST: CALL    WRNOVA          ;SEND IT TO THE NOVA
```

```
        CALL    TSTDON          ;WAIT FOR DONE TO CLEAR
        CALL    RDNOVA          ;CHECK FOR NEW COMMAND
        RET     NC              ;TAKEN IF NO NEW COMMAND
;
        JR      CPBCDE          ;TAKEN TO VALIDATE THE
                                ;NEW COMMAND
;
GETDCM: CALL    RDNOVA          ;CHECK FOR NEW DATA COMMAND
        JR      NC,GETDCM       ;TAKEN IF DATA COMMAND NOT
                                ;RECEIVED YET
CPBCDE: CP      E               ;CHECK LO BYTE OF RECEIVED
                                ;COMMAND AGAINST WHAT WAS
                                ;EXPECTED
        SCF                     ;SET CARRY TO INDICATE THAT
                                ;SOMETHING WAS RECEIVED
        RET     NZ              ;TAKEN IF NO MATCH
;
        LD      A,B             ;MOVE HI BYTE FOR TEST
        CP      D               ;CHECK HI BYTE
        SCF                     ;INDICATE RECEPTION
        RET
;
;
;       THE FOLLOWING ROUTINE IS USED BY THE DATA TRANSFER
;        ROUTINES TO TEST FOR WHEN THE END OF THE DATA
;        BUFFER HAS BEEN REACHED.  THIS ROUTINE WILL
;        RETURN WITH THE CARRY CODE SET IF THE END OF THE
;        DATA BUFFER HAS BEEN REACHED OR EXCEEDED.
;
DBETST: PUSH    BC              ;SAVE 'BC'
        PUSH    HL              ;SAVE DATA BUFFER POINTER
        DEC     HL              ;BACK UP ONE LOCATION
        PUSH    HL              ;MOVE CONTENTS OF 'HL' TO
        POP     BC              ;THE 'BC' REGISTER
        LD      HL,DATTST       ;GET ADDRESS OF BUFFER
                                ;ENDING ADDRESS
        CALL    BNDTST          ;COMPARE THE TWO
        POP     HL              ;RESTORE BUFFER POINTER
        POP     BC              ;RESTORE 'BC'
        RET
;
;
;       THE FOLLOWING ROUTINE PROVIDES THE INPUT CAPABILITY
;        FOR DATA THAT IS RECEIVED VIA MODE "01" PROGRAMMED
;        I/O.  THE RECEIVED DATA IS STORED IN MEMORY AS TWO
;        BYTES AT THE LOCATION THAT THE 'HL' REGISTER IS
;        ADDRESSING ON ENTRY.  THIS DATA IS STORED IN A
;        HI BYTE/LO BYTE FORM WITH THE HI BYTE IN THE LOWER
;        ADDRESS.  THE 'HL' REGISTER WILL BE INCREMENTED BY
;        TWO ON RETURN.
;
INPDAT: CALL    RDNOVA          ;GET A WORD FROM THE NOVA
        JR      NC,INPDAT       ;TAKEN TO LOOP UNTIL A
                                ;WORD IS RECEIVED
```

```
SAVDAT: LD      (HL),B          ;SAVE HI BYTE IN THE DATA
                                ;BUFFER
        INC     HL              ;BUMP DATA POINTER
        LD      (HL),C          ;SAVE LO BYTE IN BUFFER
        INC     HL              ;BUMP DATA POINTER
        RET
;
;
;       THE FOLLOWING ROUTINE PROVIDES THE OUTPUT CAPABILITY
;        FOR DATA THAT IS TRANSMITTED USING THE MODE "01"
;        PROGRAMMED I/O.  THE TRANSMITTED DATA IS RETRIEVED
;        FROM THE DATA BUFFER ADDRESS THAT IS CONTAINED IN
;        THE 'HL' REGISTER ON ENTRY.  THE HI BYTE OF THE
;        DATA MUST BE STORED IN THE LOWER ADDRESS.  BOTH
;        BYTES OF THE DATA WILL BE TRANSMITTED AND THE 'HL'
;        REGISTER WILL BE INCREMENTED TWICE ON RETURN.
;
OUTDAT: LD      B,(HL)          ;GET HI BYTE OF DATA
        INC     HL              ;BUMP DATA POINTER
        LD      C,(HL)          ;GET LO BYTE OF DATA
        INC     HL              ;BUMP DATA POINTER
        CALL    WRNOVA          ;SEND DATA TO NOVA
        RET
;
;
;       THE FOLLOWING ROUTINE PROVIDES THE INPUT CAPABILITY
;        FOR DATA THAT IS TO BE RECEIVED USING A MODE "10"
;        DATA CHANNEL TRANSFER.  THE RECEIVED DATA IS PUT
;        INTO THE DATA BUFFER IN THE SAME MANNER AS THE
;        PREVIOUSLY DESCRIBED "INPDAT" ROUTINE.  THIS
;        ROUTINE MUST BE ENTERED WITH THE PORT ADDRESS OF
;        THE LOW DATA BYTE IN THE 'C' REGISTER.  SINCE THE
;        CHANNEL TRANSFERS USE NOVA DISK BLOCKS (256 WORDS)
;        AS THE MEDIUM FOR MEASURING THE NUMBER OF DATA
;        WORDS TRANSFERRED, THIS ROUTINE RETURNS WITH THE
;        'B' REGISTER DECREMENTED BY ONE TO HELP COUNT THE
;        WORDS THAT HAVE BEEN TRANSFERRED.
;
INPDCH: OUT     (DCHREQ),A      ;REQUEST DCH FROM NOVA
        CALL    DCHRDY          ;WAIT FOR DCH TO COMPLETE
        INC     C               ;BUMP INPUT PORT POINTER
                                ;TO GET HI BYTE OF DATA
        INI                     ;GET NOVA HI BYTE DATA
        INC     B               ;BUMP DISK BLOCK COUNTER
                                ;SO IT WILL INDICATE
                                ;WORDS TRANSFERRED
        DEC     C               ;DECREASE INPUT PORT
                                ;POINTER FOR LO DATA BYTE
        INI                     ;GET NOVA LO BYTE DATA
        RET
;
;
```

```
;       THE FOLLOWING ROUTINE PROVIDES THE OUTPUT CAPABILITY
;        FOR DATA THAT IS TO BE TRANSMITTED TO THE NOVA
;        USING A MODE "10" DATA CHANNEL TRANSFER.  THE DATA
;        TO BE TRANSMITTED IS RETRIEVED FROM THE DATA BUFFER
;        IN THE SAME MANNER AS THE 'OUTDAT' ROUTINE.  THE
;        ENTRY REQUIREMENTS AND EXIT CONDITIONS ARE THE SAME
;        AS THOSE FOR THE 'INPDCH' ROUTINE.
;
OUTDCH: INC     C               ;BUMP OUTPUT PORT POINTER
                                ;TO SEND HI BYTE OF DATA
        OUTI                    ;PUT HI BYTE IN OUTPUT PORT
        INC     B               ;ONLY WANT TO COUNT WORDS
        DEC     C               ;POINT TO LO BYTE OUTPUT PORT
        OUTI                    ;PUT LO BYTE IN OUTPUT PORT
        CALL    DCHRDY          ;IS NOVA READY?
        OUT     (DCHREQ),A      ;REQUEST OUTPUT DCH
        RET
;
;
        END     BEGIN           ;END OF CHOPS
```

```
;*************************************************************
;
;                        NOVA/CROMEMCO
;
;                   CHANNEL OPERATING SYSTEM
;
;                        C  H  O  P  S
;
;                     MAIN COMMAND TABLE
;                           AND
;                  COMMAND INFORMATION LISTS
;                     - - - - - - - - -
;                        VERSION 2.0
;                     - - - - - - - - -
;
;                        WRITTEN BY
;
;                 CAPT GEORGE C. BEASLEY, JR., USAF
;
;                        MARCH 1981
;
;*************************************************************
;
;       THE FOLLOWING TABLE CONTAINS ALL THE INFORMATION
;        REQUIRED BY THE CHOPS FOR IT TO RECEIVE AND
;        VALIDATE COMMANDS AND PARAMETERS ASSOCIATED WITH
;        PARTICULAR TASKS.  WHEN NEW TASKS ARE ADDED THE
;        PROPER ENTRIES MUST BE MADE TO THIS TABLE TO
;        ALLOW THE CHOPS TO GAIN KNOWLEDGE OF THE NEW
;        CAPABILITY.
;
;       ENTRY   COMTBL
;
;       THE FOLLOWING GLOBAL REFERENCES ALLOW THE TABLE TO
;        BE PROPERLY FORMED BY THE LINKER.  THE SYMBOLIC
;        NAME OF EACH TASK ADDRESS ENTERED IN THE TABLE
;        MUST ALSO BE ENTERED IN AN EXTERNAL STATEMENT LIKE
;        THE FOLLOWING NAMES.
;
;       EXT     CMSTAT
;       EXT     TSK00O,TSK00I,TSK01O,TSK01I,TSK02O,TSK02I
;       EXT     TSK03O,TSK03I
;
;     THE FOLLOWING ORIGIN ADDRESS MAY NEED TO BE CHANGED
;        FOR DIFFERENT VERSIONS OF THE CHOPS; THEREFORE, THE
;        VERSION NUMBER OF THIS TABLE MUST MATCH THE VERSION
;        NUMBER OF THE MAIN SYSTEM ROUTINES.
;
```

```
;       THIS PARTICULAR TABLE WAS ESTABLISHED FOR CHOPS
;        VERSION 2.0 AND WILL HAVE TO BE ALTERED WHEN THE
;        CHOPS IS PLACED IN ROM.
;
        ORG     0B00H               ;THIS ADDRESS MUST ALWAYS
                                    ;BE SET ON A PAGE BOUNDARY
                                    ;(LO BYTE = 00)
;
;       THE FIRST 256 BYTES OF THE TABLE CONSISTS OF THE
;        POINTERS TO THE COMMAND INFORMATION LIST FOR EACH
;        VALID COMMAND.  CURRENTLY ALL OF THE COMMAND
;        INFORMATION LISTS ARE CONTAINED IN THE MAIN TABLE
;        FOLLOWING THESE POINTERS.  THE LISTS COULD BE
;        LOCATED WITHIN THE TASK ITSELF.
;
COMTBL: DW      CIL00O              ;CIL - TASK 00 - OUTPUT
        DW      CIL00I              ;CIL - TASK 00 - INPUT
        DW      CIL01O              ;CIL - TASK 01 - OUTPUT
        DW      CIL01I              ;CIL - TASK 01 - INPUT
        DW      CIL02O              ;CIL - TASK 02 - OUTPUT
        DW      CIL02I              ;CIL - TASK 02 - INPUT
        DW      CIL03O              ;CIL - TASK 03 - OUTPUT
        DW      CIL03I              ;CIL - TASK 03 - INPUT
;
;       ALL INVALID TASK NUMBERS HAVE AN ADDRESS OF ZERO
;
        DW      0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
        DW      0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
        DW      0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
        DW      0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
        DW      0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
        DW      0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
        DW      0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
        DW      0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
        DW      0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
        DW      0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
        DW      0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
        DW      0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
;
;       THE FOLLOWING ARE THE COMMAND INFORMATION LISTS
;        ASSOCIATED WITH EACH TASK.
;
CIL00O: DB      010H                ;LO NIBBLE - PARAMETER COUNT
                                    ;HI NIBBLE - DATA FLAG SET
        DW      CMSTAT              ;STATUS LOCATION FOR OUTPUT
                                    ;TASK 00
        DW      00000H              ;DATA COUNT LO BOUND
        DW      00400H              ;DATA COUNT HI BOUND
                                    ;NO ACTUAL PARAMETERS SINCE
                                    ;PARAMETER COUNT IS ZERO
                                    ;DATA COUNT INCLUDED SINCE
                                    ;DATA FLAG IS SET
        DW      TSK000              ;ADDRESS OF TASK 00
;
```

```
CIL00I: DB      10H             ;ZERO PARAMETERS - SET DATA
        DW      CMSTAT+1        ;STATUS LOCATION
        DW      00000H          ;DATA COUNT LO BOUND
        DW      00400H          ;DATA COUNT HI BOUND
        DW      TSK00I          ;TASK ADDRESS
;
CIL01O: DB      012H            ;TWO PARAMETERS - SET DATA
        DW      CMSTAT+2        ;STATUS LOCATION
        DW      00000H          ;DATA COUNT LO BOUND
        DW      00400H          ;DATA COUNT HI BOUND
        DW      00070H          ;ONE LESS THAN THE MINIMUM
                                ;SAMPLE RATE SETTING
        DW      0FFFFH          ;MAX. SAMPLE RATE SETTING
        DW      00000H          ;ONE LESS THAN THE LOWEST
                                ;A/D CHANNEL
        DW      00010H          ;HIGHEST A/D CHANNEL
        DW      TSK01O          ;TASK ADDRESS
;
CIL01I: DB      012H            ;TWO PARAMETERS - SET DATA
        DW      CMSTAT+3        ;STATUS LOCATION
        DW      00000H          ;DATA COUNT LO BOUND
        DW      00400H          ;DATA COUNT HI BOUND
        DW      00070H          ;ONE LESS THAN THE MINIMUM
                                ;SAMPLE RATE SETTING
        DW      0FFFFH          ;MAX. SAMPLE RATE SETTING
        DW      00000H          ;ONE LESS THAN THE LOWEST
                                ;D/A CHANNEL
        DW      00004H          ;HIGHEST D/A CHANNEL
        DW      TSK01I          ;TASK ADDRESS
;
CIL02O: DB      014H            ;FOUR PARAMETERS - SET DATA
        DW      CMSTAT+4        ;STATUS LOCATION
        DW      00000H          ;DATA COUNT LO BOUND
        DW      00400H          ;DATA COUNT HI BOUND
        DW      00000H          ;ONE LESS THAN THE LOWEST
                                ;PLACE TO START RETRIEVING
                                ;DATA FROM THE BUFFER
        DW      05800H          ;THE HIGHEST PLACE IN THE
                                ;DATA BUFFER FROM WHICH
                                ;DATA CAN BE RETRIEVED
        DW      00000H          ;ONE LESS THAN THE LOWEST
                                ;NUMBER OF DATA WORDS TO
                                ;BE TRANSFERRED
        DW      05800H          ;THE LARGEST NUMBER OF
                                ;DATA WORDS THAT CAN BE
                                ;TRANSFERRED
        DW      00070H          ;ONE LESS THAN THE MINIMUM
                                ;SAMPLE RATE SETTING
        DW      0FFFFH          ;MAX. SAMPLE RATE SETTING
        DW      00000H          ;ONE LESS THAN THE LOWEST
                                ;A/D CHANNEL
        DW      00010H          ;HIGHEST A/D CHANNEL
        DW      TSK02O          ;TASK ADDRESS
;
```

```
CIL02I: DB      014H            ;FOUR PARAMETERS - SET DATA
        DW      CMSTAT+5        ;STATUS LOCATION
        DW      00000H          ;DATA COUNT LO BOUND
        DW      00400H          ;DATA COUNT HI BOUND

        DW      00000H          ;ONE LESS THAN THE LOWEST
                                ;PLACE TO START RETRIEVING
                                ;DATA FROM THE BUFFER
        DW      05800H          ;THE HIGHEST PLACE IN THE
                                ;DATA BUFFER FROM WHICH
                                ;DATA CAN BE RETRIEVED
        DW      00000H          ;ONE LESS THAN THE LOWEST
                                ;NUMBER OF DATA WORDS TO
                                ;BE TRANSFERRED
        DW      05800H          ;THE LARGEST NUMBER OF
                                ;DATA WORDS THAT CAN BE
                                ;TRANSFERRED
        DW      00070H          ;ONE LESS THAN THE MINIMUM
                                ;SAMPLE RATE SETTING
        DW      0FFFFH          ;MAX. SAMPLE RATE SETTING
        DW      00000H          ;ONE LESS THAN THE LOWEST
                                ;D/A CHANNEL
        DW      00004H          ;HIGHEST D/A CHANNEL
        DW      TSK02I          ;TASK ADDRESS
;
CIL03O: DB      011H            ;ONE PARAMETER - SET DATA
        DW      CMSTAT+6        ;STATUS LOCATION
        DW      00000H          ;DATA COUNT LO BOUND
        DW      00400H          ;DATA COUNT HI BOUND
        DW      00000H          ;ONE LESS THAN THE LOWEST
                                ;SETTING OF DISPLAY TIME
        DW      0FFFFH          ;THE HIGHEST SETTING OF
                                ;THE DISPLAY TIME
        DW      TSK03O          ;TASK ADDRESS
;
CIL03I: DB      011H
        DW      CMSTAT+7
        DW      00000H          ;DATA COUNT LO BOUND
        DW      00400H          ;DATA COUNT HI BOUND
        DW      00000H          ;ONE LESS THAN THE LOWEST
                                ;SETTING OF DISPLAY TIME
        DW      0FFFFH          ;THE HIGHEST SETTING OF
                                ;THE DISPLAY TIME
        DW      TSK03I          ;TASK ADDRESS
;
;
        END                     ;END OF COMMAND TABLE
```

```
;*****************************************************************
;
;                        NOVA/CROMEMCO
;
;                    CHANNEL OPERATING SYSTEM
;
;                        C  H  O  P  S
;
;                     TASK NUMBER ZERO
;              (ENTIRE DATA BUFFER SEND AND RECEIVE)
;                     - - - - - - - - -
;                        VERSION 1.0
;                     - - - - - - - - -
;
;                        WRITTEN BY
;
;                  CAPT GEORGE C. BEASLEY, JR., USAF
;
;                        MARCH 1981
;
;*****************************************************************
;
;       THE FOLLOWING TASK IS A VERY SIMPLE ROUTINE THAT
;        CAUSES THE ENTIRE DATA BUFFER TO BE FILLED FROM
;        THE NOVA FOR AN INPUT COMMAND AND TO BE EMPTIED
;        INTO THE NOVA BY AN OUTPUT COMMAND.  THERE IS
;        NO OTHER I/O ASSOCIATED WITH THIS TASK OTHER
;        THAN THE TRANSACTIONS TO AND FROM THE NOVA.
;
;       THE FOLLOWING EXTERNAL ROUTINES AND VARIABLES ARE
;        AVAILABLE FOR USE BY THIS TASK.
;
        EXT     TCMPLT,ERROR,DATOUT,DCHOUT,DATINP,DCHINP
        EXT     DOJOBO,DOJOBI,LOMEM,HIMEM,DATSTR,DATEND
        EXT     SBUFDF
;
;
;       THE FOLLOWING DEFINITION ALLOWS THE COMMAND TABLE
;        TO PICK UP THE TASK ADDRESS DURING LINKING.  THESE
;        ENTRY POINT ADDRESSES MUST BE SPECIFIED IN THIS
;        MANNER TO ALLOW THE TASK ADDRESS TO BE INCLUDED IN
;        THE COMMAND TABLE.
;
        ENTRY   TSK000,TSK00I
;
;
ERR6    EQU     06H             ;ERROR CODE FOR
                                ;"INVALID COMMAND MODE"
;
;
```

```
;       OUTPUT TASK 00
;
TSK000: JR      OMODE0          ;MODE "00" ENTRY POINT
;
        JR      OMODE1          ;MODE "01" ENTRY POINT
;
        JR      OMODE2          ;MODE "10" ENTRY POINT
;
OMODE3: JP      TCMPLT          ;MODE "11" ENTRY POINT
;
OMODE0: LD      E,ERR6          ;MODE "00" NOT AVAILABLE
                                ;SET "INVALID COMMAND MODE"
        JP      ERROR           ;TAKEN TO HANDLE ERROR
;
OMODE1: LD      DE,DATOUT       ;GET ADDRESS OF MODE "01"
                                ;OUTPUT DATA HANDLER
        JR      GTOJOB          ;TAKEN TO CONTINUE TASK
                                ;SET UP
;
OMODE2: LD      DE,DCHOUT       ;GET ADDRESS OF MODE "10"
                                ;OUTPUT DATA HANDLER
GTOJOB: CALL    SBUFDF          ;SET BUFFER DEFAULTS
        LD      HL,JOB00        ;GET JOB ADDRESS
        JP      DOJOB0          ;LET THE CHOPS DO THE WORK
;
;
;       INPUT TASK 00
;
TSK00I: JR      OMODE0          ;MODE "00" ENTRY POINT
;
        JR      IMODE1          ;MODE "01" ENTRY POINT
;
        JR      IMODE2          ;MODE "10" ENTRY POINT
;
        JR      OMODE3          ;MODE "11" ENTRY POINT
;
IMODE1: LD      DE,DATINP       ;GET ADDRESS OF MODE "01"
                                ;INPUT DATA HANDLER
        JR      GTIJOB          ;TAKEN TO CONTINUE TASK
                                ;SET UP
;
IMODE2: LD      DE,DCHINP       ;GET ADDRESS OF MODE "10"
                                ;INPUT DATA HANDLER
GTIJOB: CALL    SBUFDF          ;SET BUFFER DEFAULTS
        LD      HL,JOB00        ;GET JOB ADDRESS
        JP      DOJOBI          ;LET THE CHOPS DO THE WORK
;
;       THIS IS ALL THERE IS OF JOB 00.  IT CONSISTS OF A
;        NULL JOB.
;
JOB00:  SCF                     ;CLEAR CARRY FLAG SO
        CCF                     ;CHOPS THINKS ALL IS OK
        RET
        END                     ;END OF TASK 00
```

```
;*************************************************************
;
;                        NOVA/CROMEMCO
;
;                   CHANNEL OPERATING SYSTEM
;
;                        C  H  O  P  S
;
;                   TASK NUMBERS ONE AND TWO
;              (A/D AND D/A COLLECTION AND TRANSFER)
;                      - - - - - - - - -
;                         VERSION 1.0
;                      - - - - - - - - -
;
;                          WRITTEN BY
;
;                  CAPT GEORGE C. BEASLEY, JR., USAF
;
;                          MARCH 1981
;
;*************************************************************
;
;       THE FOLLOWING TWO TASKS PROVIDE THE CAPABILITY OF
;        COLLECTING, TRANSFERRING, AND RECONSTRUCTING DATA
;        WITH THE A/D AND D/A CONVERTERS OF THE SYSTEM.
;
;       TASK ONE REQUIRES TWO PARAMETERS.
;         PARAMETER ONE = SAMPLING TIME INDICATOR
;         PARAMETER TWO = A/D OR D/A CHANNEL NUMBER
;
;       PARAMETER ONE IS CALCULATED FROM THE FORMULA:
;         STI = INTEGER (2x10^6/SAMPLING FREQUENCY)
;               CURRENT LIMITATIONS REQUIRE THIS NUMBER
;               TO BE GREATER THAN 199.
;
;       PARAMETER TWO SELECTS ONE OF SIXTEEN A/D CHANNELS
;        FOR THE I/O CHANNEL OUTPUT TASK OR ONE OF FOUR
;        D/A CHANNELS FOR THE I/O CHANNEL INPUT TASK.
;
;
;       TASK TWO REQUIRES FOUR PARAMETERS.
;         PARAMETER ONE   = STARTING DATA WORD FOR A/D OR D/A
;         PARAMETER TWO   = TOTAL NUMBER OF DATA WORDS FOR
;                           A/D OR D/A
;         PARAMETER THREE = SAMPLING TIME INDICATOR
;         PARAMETER FOUR  = A/D OR D/A CHANNEL NUMBER
;
;       PARAMETER ONE PROVIDES AN INDICATION TO THE TASK OF
;        THE WORD IN THE DATA BUFFER AT WHICH THE A/D OR D/A
;        CONVERSION AND/OR TRANSFER SHOULD BEGIN.
;
```

```
;       PARAMETER TWO DETERMINES THE TOTAL NUMBER OF WORDS
;        THAT WILL BE CONVERTED AND/OR TRANSFERRED.
;
;       PARAMETERS THREE AND FOUR ARE THE SAME AS PARAMETERS
;        ONE AND TWO, RESPECTIVELY, OF TASK ONE.
;
        EXT     TCMPLT,ERROR,DATOUT,DCHOUT,DATINP,DCHINP
        EXT     DOJOBO,DOJOBI,LOMEM,HIMEM,DATSTR,DATEND
        EXT     BNDTST,PRMBUF,SBUFDF,USERAM
;
;
;       THE FOLLOWING DEFINITION ALLOWS THE COMMAND TABLE
;        TO PICK UP THE TASK ADDRESS DURING LINKING.  THESE
;        ENTRY POINT ADDRESSES MUST BE SPECIFIED IN THIS
;        MANNER TO ALLOW THE TASK ADDRESS TO BE INCLUDED IN
;        THE COMMAND TABLE.
;
        ENTRY   TSK01O,TSK01I,TSK02O,TSK02I
;
;
SWTFLG: EQU     0037H           ;STORAGE ADDRESS FOR
                                ;SWITCH FLAG
IVCTOR: EQU     0038H           ;ADDRESS FOR INTERRUPT JUMP
SWTMSK: EQU     020H            ;MASK FOR EXTRACTING
                                ;SWITCH STATUS
ERR6    EQU     06H             ;ERROR CODE FOR
                                ;"INVALID COMMAND MODE"
ERR8:   EQU     08H             ;ERROR CODE FOR
                                ;"DATA BUFFER SIZE EXCEEDED"
FLAGS:  EQU     0D0H            ;STATUS FLAGS PORT
                                ;(SWITCH FLAG)
MSINTO: EQU     0E0H            ;MASTER INTERRUPT
                                ;CONTROLLER - A0=0
SLINTO: EQU     0E2H            ;SLAVE INTERRUPT
                                ;CONTROLLER - A0=0
TIMCTL: EQU     0E5H            ;TIMING CONTROLLER CONTROL
                                ;AND STATUS PORT
ATDSEL  EQU     0F8H            ;PORT FOR SELECTING ONE
                                ;OF SIXTEEN A/D INPUTS
ATDCVT  EQU     0F9H            ;PORT FOR STARTING AN
                                ;A/D CONVERSION
ATDSTS  EQU     0F9H            ;PORT FOR TESTING FOR
                                ;END OF CONVERSION
ATDHI   EQU     0FBH            ;PORT FOR READING HI
                                ;BYTE OF A/D CONVERTOR
;
;
;       TASK 01 OUTPUT
;
TSK01O: JR      OMOD10          ;MODE "00" ENTRY POINT
;
        JR      OMOD11          ;MODE "01" ENTRY POINT
;
        JR      OMOD12          ;MODE "10" ENTRY POINT
```

```
;
        JR      JOBCMP          ;MODE "11" ENTRY POINT
;
OMOD10: CALL    JOB010          ;INVOKE THE OUTPUT JOB
        JR      JOBRET          ;TAKEN FOR TASK RETURN
;
OMOD11: LD      DE,DATOUT       ;GET ADDRESS OF MODE "01"
                                ;OUTPUT DATA HANDLER
        JR      GTJB10          ;TAKEN TO CONTINUE TASK
                                ;SET UP
;
OMOD12: LD      DE,DCHOUT       ;GET ADDRESS OF MODE "10"
                                ;OUTPUT DATA HANDLER
GTJB10: CALL    SBUFDF          ;SET BUFFER DEFAULTS
        LD      HL,JOB01O       ;GET JOB ADDRESS
        JP      DOJOBO          ;LET THE CHOPS DO THE WORK
;
;
;       TASK 01 INPUT
;
TSK01I: JR      IMOD10          ;MODE "00" ENTRY POINT
;
        JR      IMOD11          ;MODE "01" ENTRY POINT
;
        JR      IMOD12          ;MODE "10" ENTRY POINT
;
        JR      JOBCMP          ;MODE "11" ENTRY POINT
;
IMOD10: CALL    JOB01I          ;INVOKE THE INPUT JOB
        JR      JOBRET          ;TAKEN FOR TASK RETURN
;
IMOD11: LD      DE,DATINP       ;GET ADDRESS OF MODE "01"
                                ;INPUT DATA HANDLER
        JR      GTJB1I          ;TAKEN TO CONTINUE TASK
                                ;SET UP
;
IMOD12: LD      DE,DCHINP       ;GET ADDRESS OF MODE "10"
                                ;INPUT DATA HANDLER
GTJB1I: CALL    SBUFDF          ;SET BUFFER DEFAULTS
        LD      HL,JOB01I       ;GET JOB ADDRESS
        JP      DOJOBI          ;LET THE CHOPS DO THE WORK
;
;
;       TASK 02 OUTPUT
;
TSK02O: JR      OMODE0          ;MODE "00" ENTRY POINT
;
        JR      OMODE1          ;MODE "01" ENTRY POINT
;
        JR      OMODE2          ;MODE "10" ENTRY POINT
;
OMODE3: JR      JOBCMP          ;MODE "11" ENTRY POINT
;
OMODE0: CALL    JOB020          ;INVOKE THE OUTPUT JOB
```

```
            JR      JOBRET          ;TAKEN FOR TASK RETURN
;
OMODE1:     LD      DE,DATOUT       ;GET ADDRESS OF MODE "01"
                                    ;OUTPUT DATA HANDLER
            JR      GTJB20          ;TAKEN TO CONTINUE TASK
                                    ;SET UP
;
OMODE2:     LD      DE,DCHOUT       ;GET ADDRESS OF MODE "10"
                                    ;OUTPUT DATA HANDLER
GTJB20:     CALL    SBUFDF          ;SET BUFFER DEFAULTS
            LD      HL,JOB020       ;GET JOB ADDRESS
            JP      DOJOBO          ;LET THE CHOPS DO THE WORK
;
;
;           TASK 02 INPUT
;
TSK02I:     JR      IMODE0          ;MODE "00" ENTRY POINT
;
            JR      IMODE1          ;MODE "01" ENTRY POINT
;
            JR      IMODE2          ;MODE "10" ENTRY POINT
;
IMODE3:     JR      JOBCMP          ;MODE "11" ENTRY POINT
;
IMODE1:     LD      DE,DATINP       ;GET ADDRESS OF MODE "01"
                                    ;INPUT DATA HANDLER
            JR      GTJB2I          ;TAKEN TO CONTINUE TASK
                                    ;SET UP
;
IMODE2:     LD      DE,DCHINP       ;GET ADDRESS OF TASK "10"
                                    ;INPUT DATA HANDLER
GTJB2I:     CALL    SBUFDF          ;SET BUFFER DEFAULTS
            LD      HL,JOB02I       ;GET JOB ADDRESS
            JP      DOJOBI          ;LET THE CHOPS DO THE WORK
;
;
IMODE0:     CALL    JOB02I          ;INVOKE THE OUTPUT JOB
JOBRET:     JP      C,ERROR         ;TAKEN IF ERROR DURING JOB
JOBCMP:     JP      TCMPLT          ;TAKEN TO WRAP UP TASK
;
;
;           JOB 01 OUTPUT
;
JOB010:     LD      A,0FFH          ;GET SWITCH ENABLE FLAG
            LD      (SWTFLG),A      ;SAVE SWITCH FLAG
            LD      HL,INTATD       ;GET ADDRESS OF A/D
                                    ;INTERRUPT SERVICE ROUTINE
            LD      (IVCTOR+1),HL   ;SET ADDRESS PORTION OF
                                    ;INTERRUPT VECTOR
            LD      HL,(PRMBUF+4)   ;GET A/D CHANNEL NUMBER
            CALL    SELATD          ;SET UP A/D CHANNEL PORT
                                    ;ADDRESS
            CALL    ENTJB1          ;INVOKE OUTPUT JOB
            JR      JOBEXT          ;TAKEN TO EXIT JOB
```

```
;
;
;       JOB 01 INPUT
;
JOB01I: LD      A,00H           ;GET SWITCH BYPASS FLAG
        LD      (SWTFLG),A      ;SAVE SWITCH FLAG
        LD      HL,INTDTA       ;GET ADDRESS OF D/A
                                ;INTERRUPT SERVICE ROUTINE
        LD      (IVCTOR+1),HL   ;SET ADDRESS PORTION OF
                                ;INTERRUPT VECTOR
        LD      HL,(PRMBUF+4)   ;GET D/A CHANNEL NUMBER
        CALL    SELDTA          ;SET UP D/A CHANNEL PORT
                                ;ADDRESS
ENTJB1: PUSH    HL              ;SAVE I/O PORT ADDRESS
        LD      HL,(DATSTR)     ;GET STARTING ADDRESS OF
                                ;DATA BUFFER
        PUSH    HL              ;SAVE DATA BUFFER START
        LD      HL,(DATEND)     ;GET ENDING ADDRESS OF
                                ;DATA BUFFER
        LD      A,L             ;GET LO BYTE OF END ADDRESS
        RRA                     ;LSB TO CARRY FOR ODD TEST
        JR      NC,SAVHL        ;TAKEN IF ENDING ADDRESS
                                ;IS EVEN
        INC     HL              ;INSURE END ADDRESS IS EVEN
SAVHL:  PUSH    HL              ;SAVE DATA BUFFER END
        LD      HL,(PRMBUF+2)   ;GET TIMING PARAMETER
        JR      JOB01           ;TAKEN TO CONTINUE JOB
;
;
;       JOB 02 OUTPUT
;
JOB02O: LD      A,0FFH          ;GET SWITCH ENABLE FLAG
        LD      (SWTFLG),A      ;SAVE SWITCH FLAG
        LD      HL,INTATD       ;GET ADDRESS OF A/D
                                ;INTERRUPT SERVICE ROUTINE
        LD      (IVCTOR+1),HL   ;SET ADDRESS PORTION OF
                                ;INTERRUPT VECTOR
        LD      HL,(PRMBUF+8)   ;GET A/D CHANNEL NUMBER
        CALL    SELATD          ;SET UP A/D PORT ADDRESS
        CALL    JOB02           ;INVOKE OUTPUT JOB
JOBEXT: RET     C               ;TAKEN IF ERROR DURING JOB
;
        LD      (DATEND),DE     ;SET DATA ENDING ADDRESS
        RET
;
;
;       JOB 02 INPUT
;
JOB02I: LD      A,00H           ;GET SWITCH BYPASS FLAG
        LD      (SWTFLG),A      ;SAVE SWITCH FLAG
        LD      HL,INTDTA       ;GET ADDRESS OF D/A
                                ;INTERRUPT SERVICE ROUTINE
        LD      (IVCTOR+1),HL   ;SET ADDRESS PORTION OF
                                ;INTERRUPT VECTOR
```

```
        LD      HL,(PRMBUF+8)   ;GET D/A CHANNEL NUMBER
        CALL    SELDTA          ;SET UP D/A PORT ADDRESS
;
JOB02:  PUSH    HL              ;SAVE I/O PORT ADDRESS
        LD      DE,(PRMBUF+2)   ;GET PARAMETER FOR DATA
                                ;BUFFER STARTING WORD
        DEC     DE              ;DECREMENT ONCE TO ADJUST
                                ;SET UP AS BYTE COUNT
        SLA     E               ;MULTIPLY REMAINING SIXTEEN
        RL      D               ;BITS BY TWO FOR BYTES
        LD      HL,(DATSTR)     ;GET ADDRESS OF BEGINNING
                                ;OF DATA BUFFER
        ADD     HL,DE           ;ADD STARTING WORD OFFSET
        PUSH    HL              ;SAVE DATA BUFFER STARTING
                                ;ADDRESS
        PUSH    HL              ;SAVE STARTING ADDRESS
                                ;FOR TRANSFER TO 'BC' REG.
        POP     BC              ;PICK UP STARTING ADDRESS
                                ;IN 'BC' REGISTER
        LD      HL,DATEND       ;GET ADDRESS OF DATA BUFFER
                                ;END POINTER
        CALL    BNDTST          ;CHECK SO BUFFER LIMIT
                                ;NOT EXCEEDED
        POP     HL              ;GET DATA BUFFER STARTING
                                ;ADDRESS IN 'HL' REGISTER
        POP     DE              ;RESTORE THE STACK IN CASE
                                ;OF ERROR RETURN
        JP      C,JOBERR        ;TAKEN IF STARTING ADDRESS
                                ;IS PAST DATA BUFFER END
        PUSH    DE              ;SAVE I/O PORT ADDRESS
        PUSH    HL              ;SAVE DATA BUFFER STARTING
                                ;ADDRESS
        LD      DE,(PRMBUF+4)   ;GET NUMBER OF WORDS TO
                                ;TRANSFER PARAMETER
        DEC     DE              ;DECREMENT ONCE TO ADJUST
                                ;FOR ADDRESSING BYTES
        SLA     E               ;MULTIPLY REMAINING SIXTEEN
        RL      D               ;BITS BY TWO FOR BYTES
        ADD     HL,DE           ;ADD DATA LENGTH TO DATA
                                ;START ADDRESS
        PUSH    HL              ;SAVE DATA END ADDRESS
        POP     BC              ;PICK UP ADDRESS OF DATA
                                ;END IN 'BC' REGISTER
        LD      HL,DATEND       ;GET ADDRESS OF DATA BUFFER
                                ;END POINTER
        CALL    BNDTST          ;CHECK TO INSURE DATA BUFFER
                                ;BOUNDS NOT EXCEEDED
        POP     HL              ;GET DATA START ADDRESS
                                ;IN 'HL' REGISTER
        POP     DE              ;RESTORE THE STACK IN CASE
                                ;OF ERROR
        JP      C,JOBERR        ;TAKEN IF DATA END ADDRESS
                                ;IS PAST DATA BUFFER END
        PUSH    DE              ;SAVE I/O PORT ADDRESS
```

```
        PUSH    HL              ;SAVE DATA START ADDRESS
        PUSH    BC              ;SAVE DATA END ADDRESS
        LD      HL,(PRMBUF+6)   ;GET TIMING PARAMETER
JOB01:  PUSH    HL              ;SAVE TIMING PARAMETER
        LD      A,0C3H          ;GET JUMP OP CODE
        LD      (IVCTOR),A      ;PUT JUMP INSTRUCTION
                                ;AT INTERRUPT ADDRESS
        LD      HL,MIOTBL       ;GET ADDRESS OF MASTER
                                ;INTERRUPT CONTROLLER SET
                                ;UP TABLE
        LD      DE,USERAM       ;GET ADDRESS OF THE USER
                                ;RAM SPACE
        LD      BC,INTATD-MIOTBL;GET SET UP TABLE LENGTH
        LDIR                    ;MOVE TABLE TO USER RAM
        EX      DE,HL           ;POINT 'HL' TO TABLE END
        POP     DE              ;GET TIMING PARAMETER
        DEC     HL              ;POSITION TABLE POINTER TO
        DEC     HL              ;ENTRY FOR TIMING PARAMETER
        LD      (HL),D          ;SAVE HI BYTE
        DEC     HL              ;BUMP TABLE POINTER
        LD      (HL),E          ;SAVE LO BYTE
        LD      B,04H           ;GET COUNT FOR MASTER
                                ;INTERRUPT CONTROLLER SET UP
        LD      HL,USERAM       ;POINT TO TOP OF SET UP TABLE
        LD      C,MSINTO        ;MASTER INTERRUPT CONTROLLER
                                ;PORT ADDRESS
        OUTI                    ;SET UP MASTER CONTROLLER
        INC     C               ;BUMP PORT ADDRESS
        OTIR                    ;COMPLETE MASTER SET UP
        LD      B,04H           ;GET COUNT FOR SLAVE
        INC     C               ;BUMP PORT ADDRESS TO SLAVE
        OUTI                    ;SET UP SLAVE FROM TABLE
        INC     C               ;BUMP PORT ADDRESS
        OTIR                    ;COMPLETE SLAVE SET UP
        LD      B,02H           ;GET COUNT FOR TIMING
                                ;CONTROLLER SET UP
        LD      C,TIMCTL        ;GET TIMING CONTROLLER
                                ;PORT ADDRESS
        OTIR                    ;SET UP TIMING CONTROLLER
        LD      B,04H           ;GET NEXT SET UP COUNT
        DEC     C               ;DECREASE PORT ADDRESS
        OTIR                    ;CONTINUE SET UP FOR TABLE
        INC     C               ;BUMP PORT ADDRESS
        OUTI                    ;COMPLETE TIMING CONTROLLER
                                ;SET UP
        POP     DE              ;GET DATA END ADDRESS
        POP     HL              ;GET DATA START ADDRESS
        POP     BC              ;GET I/O PORT ADDRESS
        IM      1               ;SET 8080 INTERRUPT MODE
        LD      A,(SWTFLG)      ;GET DECISION SWITCH FLAG
        OR      A               ;SET CONDITION CODES
        JR      Z,SRTINT        ;TAKEN IF SWITCH NOT NEEDED
SWITCH: IN      A,FLAGS         ;INPUT NOVA FLAGS
        AND     SWTMSK          ;EXTRACT THE SWITCH DATA
```

```
        JR      NZ,SWITCH       ;TAKEN IF SWITCH NOT
                                ;PRESSED
SRTINT: EI                      ;INTERRUPTS OK NOW
HILOOP: LD      A,H             ;GET HI BYTE OF DATA
                                ;ADDRESS
        CP      D               ;CHECK IF AT END YET
        JR      C,HILOOP        ;TAKEN IF NOT ON LAST
                                ;PORTION
LOLOOP: LD      A,L             ;GET LO BYTE OF DATA ADDRESS
        CP      E               ;CHECK IF LAST BYTE CONVERTED
        JR      C,LOLOOP        ;TAKEN IF NOT FINISHED
        DI                      ;FINISHED WITH INTERRUPTS
        LD      A,0FFH          ;GET TIMER RESET CODE
        OUT     (TIMCTL),A      ;RESET THE TIMING CONTROLLER
        RET
;
JOBERR: LD      E,ERR8          ;GET ERROR CODE FOR
                                ;"DATA BUFFER SIZE EXCEEDED"
        RET
;
;
;       THE FOLLOWING SUBROUTINE SELECTS ONE OF SIXTEEN
;        OF THE ANALOG TO DIGITAL CONVERTER.  THE DESIRED
;        CHANNEL IS PASSED IN THE LEAST SIGNIFICANT FOUR
;        BITS OF THE 'L' REGISTER.  AFTER SELECTING THE
;        CHANNEL, THE ROUTINE RETURNS WITH THE PORT ADDRESS
;        OF THE HIGH BYTE OF A/D DATA IN THE 'L' REGISTER.
;        THE 'H' REGISTER IS ZEROED ON RETURN.
;
SELATD: LD      A,L             ;GET A/D CHANNEL NUMBER
        OUT     (ATDSEL),A      ;SET A/D CHANNEL NUMBER
        LD      HL,ATDHI        ;PUT ADDRESS OF A/D HI DATA
                                ;BYTE IN 'L' REGISTER
        RET
;
;
;       THE FOLLOWING SUBROUTINE GENERATES THE PROPER DIGITAL
;        TO ANALOG CONVERTER PORT ADDRESS BASED ON THE NUMBER
;        PASSED IN THE LEAST SIGNIFICANT TWO BITS OF THE 'L'
;        REGISTER.  THE GENERATED PORT ADDRESS IS RETURNED IN
;        THE 'L' REGISTER.
;
;       THE MAPPING BETWEEN THE D/A PORT ADDRESSES AND THE
;        NUMBER THAT IS PASSED DOES NOT DIRECTLY CORRESPOND
;        TO THE NUMBERING OF THE PORT ADDRESSES IN THE
;        DOCUMENTATION FOR THE D/A CONVERTER.  TO SIMPLIFY
;        THIS ROUTINE THE FOLLOWING MAPPING HAS BEEN USED:
;
;             CHOPS                   DOCUMENTED
;               1                         B
;               2                         C
;               3                         D
;               4                         A
;
```

```
SELDTA: LD      A,L             ;GET D/A CONVERTER NUMBER
        SLA     A               ;MULTIPLY BY TWO
        AND     07H             ;GET LOWER THREE BITS
        OR      0F0H            ;SET FOUR HIGHER BITS TO
                                ;GENERATE PORT ADDRESS

        LD      L,A             ;PUT IN 'L' REGISTER
        RET
;
;
;       THE FOLLOWING ROUTINE PROVIDES THE ACKNOWLEDGE TO THE
;        MASTER AND SLAVE INTERRUPT CONTROLLERS.
;
INTACK: LD      A,00CH          ;CONTROLLER OCW3 WITH POLL
                                ;BIT SET
        OUT     (MSINTO),A      ;POLL MASTER INTERRUPT
                                ;CONTROLLER
        OUT     (SLINTO),A      ;POLL SLAVE INTERRUPT
                                ;CONTROLLER
        IN      A,(MSINTO)      ;ACKNOWLEDGE INTERRUPT AT
                                ;THE MASTER
        IN      A,(SLINTO)      ;ACKNOWLEDGE INTERRUPT AT
                                ;THE SLAVE
        RET
;
;
;       THE FOLLOWING TABLE CONTAINS THE VALUES NEEDED TO SET
;        UP THE INTERRUPT AND TIMING CONTROLLERS.  THE TABLE
;        CONTENTS ARE TRANSFERRED TO THE CHOPS USER RAM AREA
;        AND THE TIMING PARAMETER IS INSERTED IN THE TABLE BY
;        THE ROUTINES WITHIN THE TASK.  THE TABLE VALUES ARE
;        THEN TRANSFERRED TO THE PROPER REGISTERS OF THE
;        RESPECTIVE CONTROLLER TO COMPLETE THE SET UP.  THE
;        DOCUMENTATION FOR EACH OF THE CONTROLLERS SHOULD BE
;        REFERENCED TO OBTAIN A DETAILED DESCRIPTION OF THE
;        CONTROLLERS' OPERATION AND SET UP VALUES.
;
MIOTBL: DB      011H            ;ICW1   - A5,A6,A7 = 0
                                ;       - EDGE TRIGGERED
                                ;         INPUT MODE
                                ;       - EIGHT BYTE ENTRIES
                                ;         IN INTERRUPT TABLE
MI1TBL: DB      000H            ;ICW2   - A8 THRU A15 = 0
        DB      002H            ;ICW3   - SELECT MASTER MODE
        DB      00EH            ;ICW4   - NON-SPECIAL FULLY
                                ;         NESTED MODE
                                ;       - MASTER ID
                                ;       - AUTO END OF
                                ;         INTERRUPT
SLOTBL: DB      011H            ;ICW1   - A5,A6,A7 = 0
                                ;       - EDGE TRIGGERED
                                ;         INPUT MODE
                                ;       - EIGHT BYTE ENTRIES
                                ;         IN INTERRUPT TABLE
```

```
        DB      000H            ;ICW2   - A8 THRU A15 = 0
        DB      001H            ;ICW3   - SELECT SLAVE MODE
        DB      00AH            ;ICW4   - NON-SPECIAL FULLY
                                ;         NESTED MODE
                                ;       - SLAVE ID
                                ;       - AUTO END OF
                                ;         INTERRUPT
TCRTBL: DB      0FFH            ;TIMING CONTROLLER MASTER
                                ;RESET CODE
        DB      003H            ;DATA POINTER REGISTER #3
        DW      0B22H           ;COUNTER MODE WITH
                                ;       - TC TOGGLE
                                ;       - COUNT DOWN
                                ;       - BINARY COUNT
                                ;       - REPETITIVELY
                                ;       - 4 MHZ TIME BASE
COUNT:  DW      0000H           ;DIVISOR FOR LOAD REGISTER
                                ;TIMING PARAMETER GOES HERE
        DB      064H            ;LOAD AND ARM COUNTER #3
;
;
;       THE FOLLOWING ROUTINE PROVIDES THE INTERRUPT SERVICE
;        FOR THE ANALOG TO DIGITAL CONVERTER.  THE ROUTINE
;        INITIATES A CONVERSION, ACKNOWLEDGES THE INTERRUPT
;        COLLECTS THE DATA AND STORES IT IN THE BUFFER.  IT
;        REQUIRES THE 'HL' REGISTER TO BE SET TO THE ADDRESS
;        IN WHICH THE DATA IS TO BE PLACED, THE 'C' REGISTER
;        MUST CONTAIN THE PORT ADDRESS OF THE HIGH DATA BYTE.
;        THE 'HL' REGISTER WILL BE INCREMENTED BY TWO BEFORE
;        THE ROUTINE RETURNS.  THE OTHER REGISTERS ARE NOT
;        CHANGED.
;
INTATD: OUT     (ATDCVT),A      ;START A CONVERSION
        DI                      ;DISABLE POSSIBLE INTERRUPT
        PUSH    AF              ;SAVE ACCUMULATOR AND STATUS
        CALL    INTACK          ;ACKNOWLEDGE INTERRUPT
ATDTST: IN      A,(ATDSTS)      ;GET A/D STATUS
        RRA                     ;READY FLAG TO CARRY
        JP      NC,ATDTST       ;TAKEN IF CONVERSION
                                ;NOT COMPLETE
        INI                     ;GET AND STORE HI DATA BYTE
        DEC     C               ;SET PORT ADDRESS TO LO
                                ;DATA BYTE
        INI                     ;GET AND STORE LO DATA BYTE
        INC     C               ;SET PORT ADDRESS TO HI
                                ;DATA BYTE
        JP      INTRET          ;TAKEN TO COMPLETE
                                ;INTERRUPT SERVICE
;
;
;       THE FOLLOWING ROUTINE PROVIDES THE INTERRUPT SERVICE
;        FOR THE DIGITAL TO ANALOG CONVERTER.  THE ROUTINE
;        ACKNOWLEDGES THE INTERRUPT AND RETRIEVES THE DATA
;        FROM THE BUFFER.  IT REQUIRES THE 'HL' REGISTER TO
```

```
;         BE SET TO THE ADDRESS IN WHICH THE DATA IS STORED,
;         THE 'C' REGISTER MUST CONTAIN THE PORT ADDRESS OF
;         THE HIGH DATA BYTE.  THE 'HL' REGISTER WILL BE
;         INCREMENTED BY TWO BEFORE THE ROUTINE RETURNS.
;         THE OTHER REGISTERS ARE NOT CHANGED.
;
INTDTA: DI                          ;DISABLE POSSIBLE INTERRUPT
        PUSH    AF                  ;SAVE ACCUMULATOR AND STATUS
        CALL    INTACK              ;ACKNOWLEDGE INTERRUPT
        OUTI                        ;SET HI DATA BYTE IN D/A
        INC     C                   ;SET PORT ADDRESS TO
                                    ;LO DATA BYTE
        OUTI                        ;SET LO DATA BYTE IN D/A
        DEC     C                   ;SET PORT ADDRESS TO
                                    ;HI DATA BYTE
INTRET: LD      A,020H              ;GET OCW2 WITH NON-SPECIFIC
                                    ;END OF INTERRUPT
        OUT     (MSINTO),A          ;CLEAR INTERRUPT ON MASTER
        OUT     (SLINTO),A          ;CLEAR INTERRUPT ON SLAVE
        POP     AF                  ;RESTORE ACCUMULATOR AND
                                    ;STATUS
        EI                          ;OK FOR MORE INTERRUPTS
        RETI
;
;
        END                         ;END OF TASKS ONE AND TWO
```

```
;*************************************************************
;                                                            ;
;                        NOVA/CROMEMCO                       ;
;                                                            ;
;                   CHANNEL OPERATING SYSTEM                 ;
;                                                            ;
;                        C  H  O  P  S                       ;
;                                                            ;
;                     TASK NUMBER THREE                      ;
;          (VIDEO DATA DISPLAY, COLLECTION, AND TRANSFER)    ;
;                     - - - - - - - - -                      ;
;                        VERSION 1.0                         ;
;                     - - - - - - - - -                      ;
;                                                            ;
;                        WRITTEN BY                          ;
;                                                            ;
;                 CAPT GEORGE C. BEASLEY, JR., USAF          ;
;                                                            ;
;                        MARCH 1981                          ;
;                                                            ;
;*************************************************************
;
;       THE FOLLOWING TASK PROVIDES THE CAPABILITY TO COLLECT
;        AND DISPLAY VIDEO INFORMATION.  THE INFORMATION THAT
;        IS DISPLAYED MAY EITHER COME FROM THE NOVA OR FROM
;        AN EXTERNAL CAMERA CONNECTED TO THE SYSTEM.
;        SELECTING INPUT OR OUTPUT WITH THE VARIOUS DATA
;        TRANSFER MODES, PROVIDES THESE POSSIBILITIES OF THE
;        SOURCE OF THE DISPLAYED VIDEO DATA.
;
;       UTILIZATION OF THIS TASK REQUIRES THAT THE THREE
;        TECMAR VIDEO DIGITIZER BOARDS WITH SUITABLE VIDEO
;        MONITOR AND CAMERA CONNECTED BE PROPERLY INSTALLED
;        IN THE CROMEMCO.
;
;       THIS TASK REQUIRES ONE PARAMETER TO BE PASSED TO IT.
;        THIS PARAMETER REPRESENTS THE APPROXIMATE TIME IN
;        SECONDS TO DISPLAY THE VIDEO INFORMATION BEFORE
;        COLLECTION AND/OR TRANSFER TO THE NOVA OR AFTER
;        TRANSFER FROM THE NOVA.
;
;
;       THE FOLLOWING EXTERNAL ROUTINES AND VARIABLES ARE
;        AVAILABLE FOR USE BY THIS TASK.
;
;       EXT     TCMPLT,ERROR,DATOUT,DCHOUT,DATINP,DCHINP
;       EXT     DOJOBO,DOJOBI,LOMEM,HIMEM,DATSTR,DATEND
;       EXT     BNDTST,PRMBUF
;
;
```

```
;       THE FOLLOWING DEFINITION ALLOWS THE COMMAND TABLE
;        TO PICK UP THE TASK ADDRESS DURING LINKING.  THESE
;        ENTRY POINT ADDRESSES MUST BE SPECIFIED IN THIS
;        MANNER TO ALLOW THE TASK ADDRESS TO BE INCLUDED IN
;        THE COMMAND TABLE.
;
        ENTRY   TSK030,TSK03I
;
;
IVCTOR: EQU     0038H           ;ADDRESS FOR INTERRUPT VECTOR
                                ;JUMP STORAGE
ERR6    EQU     06H             ;ERROR CODE FOR
                                ;"INVALID COMMAND MODE"
ERR8:   EQU     08H             ;ERROR CODE FOR
                                ;"DATA BUFFER SIZE EXCEEDED"
VIDCTL: EQU     0FFH            ;TIMING CONTROLLER
                                ;CONTROL AND STATUS PORT
;
;
;       OUTPUT TASK 03
;
TSK030: JR      OMODE0          ;MODE "00" ENTRY POINT
;
        JR      OMODE1          ;MODE "01" ENTRY POINT
;
        JR      OMODE2          ;MODE "10" ENTRY POINT
;
        JP      TCMPLT          ;MODE "11" ENTRY POINT
;
OMODE0: CALL    SETBUF          ;SET UP BUFFER ADDRESSES
        JR      C,JOBERR        ;TAKEN IF ERROR IN SETTING
                                ;UP VIDEO DATA BUFFER
        CALL    JOB030          ;INVOKE THE OUTPUT JOB
        JR      JOBRET          ;TAKEN FOR ORDERLY RETURN
                                ;TO THE CHOPS
;
OMODE1: LD      DE,DATOUT       ;GET ADDRESS OF MODE "01"
                                ;OUTPUT DATA HANDLER
        JR      GTOJOB          ;TAKEN TO CONTINUE TASK
;
OMODE2: LD      DE,DCHOUT       ;GET ADDRESS OF MODE "10"
                                ;OUTPUT DATA HANDLER
GTOJOB:CALL     SETBUF          ;SET UP BUFFER ADDRESSES
        JR      C,JOBERR        ;TAKEN IF ERROR IN SETTING
                                ;UP VIDEO DATA BUFFER
        LD      HL,JOB030       ;GET ADDRESS OF OUTPUT JOB
        JP      DOJOBO          ;LET THE CHOPS DO THE WORK
;
;
;       INPUT TASK 03
;
TSK03I: JR      IMODE0          ;MODE "00" ENTRY POINT
;
        JR      IMODE1          ;MODE "01" ENTRY POINT
```

```
;
        JR      IMODE2          ;MODE "10" ENTRY POINT
;
        JP      TCMPLT          ;MODE "11" ENTRY POINT
;
IMODE1: LD      DE,DATINP       ;GET ADDRESS OF MODE "01"
                                ;INPUT DATA HANDLER
        JR      GTIJOB          ;TAKEN TO CONTINUE TASK
;
IMODE2: LD      DE,DCHINP       ;GET ADDRESS OF MODE "10"
                                ;INPUT DATA HANDLER
GTIJOB: CALL    SETBUF          ;SET UP BUFFER ADDRESSES
        JR      C,JOBERR        ;TAKEN IF ERROR IN SETTING
                                ;UP VIDEO DATA BUFFER
        LD      HL,JOB03I       ;GET ADDRESS OF INPUT JOB
        JP      DOJOBI          ;LET THE CHOPS DO THE WORK
;
IMODE0: LD      BC,(LOMEM)      ;GET ADDRESS OF LOWEST DATA
                                ;BUFFER LOCATION
        LD      HL,(DATSTR)     ;GET ADDRESS OF FIRST DATA
                                ;WORD IN BUFFER
        CALL    BNDTST          ;CHECK FOR PROPER PICTURE
                                ;STARTING ADDRESS
        JR      C,JOBERR        ;TAKEN IF BEGINNING OF
                                ;PICTURE NOT POSITIONED
                                ;PROPERLY IN BUFFER
        CALL    JOB03I          ;DISPLAY INPUT PICTURE
JOBRET: JP      NC,TCMPLT       ;TAKEN IF NO JOB ERROR
JOBERR: LD      E,ERR8          ;GET ERROR CODE FOR
                                ;"DATA BUFFER SIZE EXCEEDED"
        JP      ERROR           ;TAKEN TO HANDLE ERROR
;
;
;       THE FOLLOWING JOB SETS UP THE VIDEO DATA BUFFER,
;        STARTS THE A/D AND D/A OF THE VIDEO DIGITIZER
;        (DISPLAYS THE DIGITIZED IMAGE WHILE COLLECTING),
;        AND DELAYS FOR A SPECIFIED TIME.
;
JOB030: LD      HL,(DATSTR)     ;GET VIDEO DATA BUFFER
                                ;STARTING ADDRESS
        CALL    SETVID          ;GENERATE DIGITIZER COMMAND
        RET     C               ;TAKEN IF ERROR IN COMMAND
;
        OR      0C0H            ;SET COMMAND BYTE TO COLLECT
        OUT     (VIDCTL),A      ;COMMAND DIGITIZER
        CALL    DISVID          ;DISPLAY THE VIDEO IMAGE
        SCF                     ;CLEAR CARRY FLAG FOR NO
        CCF                     ;ERROR RETURN TO THE CHOPS
        RET
;
;
;       THE FOLLOWING JOB SETS UP THE VIDEO DATA BUFFER,
;        STARTS THE D/A OF THE VIDEO DIGITIZER AND DELAYS
;        FOR A SPECIFIED TIME.
```

```
;
JOB03I: LD      HL,(DATSTR)     ;GET VIDEO DATA BUFFER
                                ;STARTING ADDRESS
        CALL    SETVID          ;GENERATE DIGITIZER COMMAND
        RET     C               ;TAKEN IF ERROR IN COMMAND
;
        OR      080H            ;SET COMMAND BYTE TO DISPLAY
        OUT     (VIDCTL),A      ;COMMAND DIGITIZER
DISVID: LD      HL,(PRMBUF+2)   ;GET TIMING PARAMETER
TLOOP:  LD      BC,1100H        ;SET UP COUNT FOR ONE
                                ;SECOND TIMING LOOP
TLOOP1: DEC     C               ;LOOP FOR APPROXIMATELY
        JR      NZ,TLOOP1
        DEC     B               ;ONE SECOND
        JR      NZ,TLOOP1
        DEC     HL              ;DECREMENT SECOND COUNT
        LD      A,L             ;GET LO BYTE OF TIME
        CP      00H             ;CHECK FOR ZERO
        JR      NZ,TLOOP        ;TAKEN TO WAIT A SECOND
        LD      A,H             ;GET HI BYTE OF TIME
        CP      00H             ;CHECK FOR ZERO
        JR      NZ,TLOOP        ;TAKEN FOR ANOTHER SECOND
        XOR     A               ;SET DIGITIZER STOP COMMAND
        OUT     (VIDCTL),A      ;COMMAND DIGITIZER
        RET
;
;
;       THE FOLLOWING SUBROUTINE SETS UP THE VIDEO DATA
;        BUFFER AND INSURES THAT THERE IS ENOUGH MEMORY
;        AVAILABLE IN THE BUFFER TO DISPLAY A COMPLETE
;        VIDEO IMAGE.  IF THERE IS NOT ENOUGH MEMORY THIS
;        ROUTINE WILL RETURN WITH THE CARRY FLAG SET.
;
SETBUF: PUSH    DE              ;SAVE THE 'DE' REGISTER
        LD      HL,(LOMEM)      ;GET LOWEST DATA BUFFER
                                ;STARTING ADDRESS
        LD      (DATSTR),HL     ;SET DATA START ADDRESS
        LD      DE,07FFFH       ;GET VIDEO DATA LENGTH
        ADD     HL,DE           ;FIND END ADDRESS
        PUSH    HL              ;TRANSFER 'HL' REGISTER
        POP     BC              ;TO THE 'BC' REGISTER
        LD      HL,(HIMEM)      ;GET HIGHEST DATA BUFFER
                                ;ENDING ADDRESS
        CALL    BNDTST          ;CHECK FOR ENOUGH ROOM
        POP     DE              ;RESTORE OLD 'DE' REGISTER
        RET     C               ;CARRY SET ON ERROR
        LD      (DATEND),BC     ;SAVE DATA END ADDRESS
        RET
;
;
;       THE FOLLOWING ROUTINE GENERATES THE ADDRESS PORTION
;        OF THE DIGITIZER COMMAND BYTE.  THIS ROUTINE USES
;        THE ADDRESS PASSED IN THE 'HL' REGISTER TO SET THE
;        PROPER BITS OF THE COMMAND BYTE SO THE DIGITIZER
```

```
;         WILL KNOW WHERE IN MEMORY THE VIDEO DATA SHOULD BE.
;         THIS ROUTINE ALSO CHECKS TO INSURE THAT THE VIDEO
;         DATA STARTS ON AN EVEN PAGE BOUNDARY.  THIS IS AN
;         ABSOLUTE CONSTRAINT BASED ON THE LIMITATIONS OF THE
;         DIGITIZER.  IF VIDEO DATA DOES NOT START ON AN EVEN
;         BOUNDARY THIS ROUTINE RETURNS WITH THE CARRY FLAG
;         SET.
;
SETVID: LD      A,L             ;GET LO BYTE OF DATA ADDRESS
        OR      A               ;SET CONDITION CODES
        JR      Z,PAGEOK        ;TAKEN IF EVEN PAGE BOUNDARY
        SCF                     ;SET CARRY FLAG FOR ERROR
        RET
;
PAGEOK: LD      A,H             ;GET HI BYTE OF ADDRESS
        SRL     A               ;PROPERLY POSITION BITS IN
        SRL     A               ;THE COMMAND BYTE
        RET                     ;CARRY WILL BE CLEAR IF
                                ;VALID STARTING ADDRESS
;
;
        END                     ;END OF TASK 03
```

Appendix D

Program Listing - CHANNEL

```
C++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++C
C                                                               C
C                        NOVA/CROMEMCO                          C
C                                                               C
C                     I/O CHANNEL DRIVER                        C
C                                                               C
C                                                               C
C                     C  H  A  N  N  E  L                       C
C                                                               C
C                                                               C
C                         VERSION 1.1                           C
C                                                               C
C                     - - - - - - - - - -                       C
C                                                               C
C                         WRITTEN  BY                           C
C                                                               C
C                     CAPT DAN FREDAL, USAF                     C
C                                                               C
C                          MARCH 1981                           C
C                                                               C
C++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++C
C
C       CHANNEL  IS  A  FORTRAN  SUBROUTINE.  THE  FOLLOWING
C   PROCEDURE  MUST BE FOLLOWED TO CREATE OBJECT CODE  WHICH
C   INCLUDES CHANNEL.  CHANNEL MUST BE COMPILED USING THE DG
C   FORTRAN IV COMPILER AND THEN THE RELOCATABLE BINARY CODE
C   MUST  BE  LINKED  WITH THE RELOCATABLE  CODE  OF  SANDS,
C   CANDR,  DCHTX,  DCHRX, AND FORT.LB.  THIS SHOULD BE DONE
C   USING RLDR.  CHANNEL MUST BE CALLED AS FOLLOWS:
C
C       CALL CHANNEL(ITASK, DIR, MODE, PCNT, DCOUNT, FILENAM,
C                    DCHBLKS, DARRAY, PARRAY, ERROR, SYSERR)
C
C       NOTE:   ALL  VARIABLES  AND ARRAYS LISTED BELOW  ARE
C               INTEGER.
C
C       ITASK-   PASSED  TO CHANNEL.   THIS IS THE NUMBER OF
C                THE TASK WHICH IS TO BE EXECUTED BY THE I/O
C                CHANNEL.
C
C       DIR-     PASSED TO CHANNEL.   THIS IS THE  DIRECTION
C                IN  WHICH ANY DATA WILL  FLOW WHEN THE TASK
C                IS EXECUTED.
C
C       MODE-    PASSED TO CHANNEL.  THIS SPECIFIES THE MODE
C                OF THE TASK TO BE EXECUTED.
C
C       PCNT-    PASSED  TO  CHANNEL.   THIS  SPECIFIES  THE
C                NUMBER  OF PARAMETERS  TO  BE PASSED TO THE
C                I/O CHANNEL.
C
C                RETURN FROM CHANNEL.  SHOULD AN ERROR OCCUR
C                DURING  THE  TRANSFER  OF  PARAMETERS,  THE
```

```
C                NUMBER OF THE PARAMETER BEING PASSED IS
C                RETURNED IN THIS ARGUMENT.
C
C        DCOUNT- PASSED TO CHANNEL. THIS SPECIFIES THE
C                NUMBER OF DATA WORDS TO BE TRANSFERRED
C                DURING A MODE 1 TASK AND THE NUMBER OF DISK
C                BLOCKS IN EACH DATA CHANNEL (DCH) BLOCK FOR
C                A MODE 2 TASK.
C
C                RETURNED FROM CHANNEL. SHOULD AN ERROR
C                OCCUR DURING THE TRANSFER OF DATA FOR A
C                MODE 1 TASK, THE NUMBER OF THE DATA WORD
C                BEING PASSED AT THE TIME IS RETURNED IN
C                THIS ARGUMENT.
C
C        FILENAM- PASSED TO CHANNEL. THIS IS A CHARACTER
C                ARRAY WHICH CONTAINS THE NAME OF THE FILE
C                INVOLVED IN A MODE 2 TASK. THE ARRAY CAN
C                CONTAIN UP TO 14 CHARACTERS (INCLUDING
C                PUNCTUATION) AND THE DIRECTORY MAY BE PUT
C                ON AS A PREFIX.
C
C        DCHBLKS- PASSED TO CHANNEL. IF THIS ARGUMENT IS
C                ZERO FOR AN INPUT MODE 2 TASK, A RANDOM
C                FILE IS CREATED. IF IT CONTAINS AN INTEGER
C                OTHER THAN ZERO, A CONTIGUOUS FILE OF THAT
C                SIZE (IN BLOCKS) WILL BE CREATED.
C
C        DARRAY- PASSED TO CHANNEL. THIS ARRAY CONTAINS THE
C                DATA TO BE OUTPUT TO A MODE 1 TASK. THE
C                NUMBER OF DATA WORDS CONTAINED IN THE ARRAY
C                MUST AGREE WITH DCOUNT. THE MEMORY FOR
C                THIS ARRAY IS DYNAMICALLY ALLOCATED AND IS
C                THEREFORE SUBJECT TO THE LIMITATIONS
C                ASSOCIATED WITH THIS PROCEDURE.
C
C                RETURNED FROM CHANNEL. THIS ARRAY WILL
C                CONTAIN THE DATA RECEIVED DURING A MODE 1
C                TASK. THE SAME MEMORY CONSTRAINTS AS ABOVE
C                ALSO APPLY.
C
C        PARRAY- PASSED TO CHANNEL. THIS ARRAY CONTAINS THE
C                PARAMETERS TO BE PASSED TO THE I/O CHANNEL.
C                THE NUMBER OF PARAMETERS MUST AGREE WITH
C                PCOUNT.
C
C        ERROR-  RETURNED FROM CHANNEL. SHOULD AN ERROR
C                OCCUR DURING THE I/O TRANSACTIONS IT WILL
C                BE RETURNED HERE. ERROR IS DIVIDED INTO
C                TWO FIELDS. THE LEAST SIGNIFICANT BYTE
C                WILL CONTAIN THE ERROR RETURNED BY THE I/O
C                CHANNEL. THE MOST SIGNIFICANT BYTE WILL
C                CONTAIN THE CHANNEL ERROR, IF ANY. IF NO
C                ERROR OCCURES A ZERO WILL BE RETURNED.
```

```
C
C       SYSERR-  RETURNED  FROM CHANNEL.   ANY SYSTEM  ERROR
C                WHICH OCCURS DURING AN I/O TRANSACTION WILL
C                BE  RETURNED HERE.   IT WILL BE IN THE FORM
C                OF A FORTRAN SYSTEM ERROR.   A ONE WILL  BE
C                RETURNED  IF NO ERROR OCCURS.   ALL  SYSTEM
C                ERRORS  WILL  HAVE  AN  ASSOCIATED  CHANNEL
C                ERROR EXCEPT AN END OF FILE ERROR (9).
C
C         CHANNEL  PROVIDES  A SOFTWARE INTERFACE TO THE  I/O
C    CHANNEL.   THIS  CHANNEL CONSISTS OF A CROMEMCO MICRO-
C    COMPUTER  WITH THE CHOPS OPERATING SYSTEM INSTALLED  AND
C    RUNNING   IN IT.   CHANNEL INITIATES TASKS AND  TRANSFERS
C    DATA  ACCORDING  TO  THE  PROTOCOL  SETUP  FOR  THE  I/O
C    CHANNEL.   IT  SHOULD  BE  NOTED THAT IF  CHOPS  IS  NOT
C    RUNNING,  CHANNEL  WILL HANG AND CTRL A MUST BE USED  TO
C    RETURN TO THE CLI.
C
C
C
C                    *** CHANNEL VARIABLES ***
C
C       ABRTERR-  ABORT ERROR.   USED AS BOTH A VARIABLE AND
C                 A  FLAG,  MOST SIGNIFICANT BIT IS SET WHEN
C                 ABORT IS INITIATED.
C
C      CMD-       COMMAND. USED TO SEND I/O CHANNEL COMMANDS
C
C       DARRAY-   SAME AS ABOVE.
C
C       DCHBLKS-  SAME AS ABOVE.
C
C       DCMD-     DATA COMMAND.  REQUIRED BY THE I/O CHANNEL
C                 PROTOCOL.
C
C       EOF-      END OF FILE.   USED TO DETECT A SYSTEM EOF
C                 ERROR (9).
C
C       ERRBIT-   ERROR BIT.  USED TO POINT TO THE ERROR BIT
C                 IN THE I/O CHANNEL'S STATUS WORD.
C
C       ERRMSK-   ERROR  MASK.   USED  TO SEPARATE  OUT  THE
C                 ERROR NIBBLE FROM THE I/O CHANNEL'S STATUS
C                 WORD.
C
C       ERROR-    SAME AS ABOVE.
C
C       FILENAM-  SAME AS ABOVE.
C
C       IER-      USED  TO COLLECT ANY ERROR RETURNED  BY  A
C                 SYSTEM CALL.
C
C       IMODE-    USED  TO STORE THE MODE WITH OFFSET OF ONE
C                 FOR COMPUTED GOTO.
```

```
C
C       IPARRAY-  CONTAINS  WORKING  VERSION  OF  PARRAY  SO
C                 CHANGES CAN BE MADE WITHOUT AFFECTING  THE
C                 ORIGINAL ARRAY.
C
C       ISTAT-    THIS  ARRAY IS USED TO STORE THE STATUS OF
C                 FILENAM OBTAINED BY CALLING STATUS.
C
C       ITASK-    SAME AS ABOVE.
C
C       MODE-     SAME AS ABOVE.
C
C       MSB-      MOST  SIGNIFICANT BIT.   USED TO POINT  TO
C                 THE MOST SIGNIFICANT BIT OF A WORD.
C
C       NOABRT-   UNABLE TO ABORT ERROR CODE.
C
C       PARRAY-   SAME AS ABOVE.
C
C       PCNT-     SAME AS ABOVE.
C
C       PCNTMSK-  PARAMETER COUNT MASK.  USED TO EXTRACT THE
C                 PARAMETER  COUNT  FROM THE  I/O  CHANNEL'S
C                 STATUS WORD.
C
C       PCOUNT-   SAME AS ABOVE.
C
C       RDOSERR-  RDOS  ERROR.   USED TO RETURN RDOS  SYSTEM
C                 ERRORS.
C
C       STATBIT-  STATUS BIT.   USED TO POINT TO THE  STATUS
C                 BIT OF THE I/O CHANNEL STATUS WORD.
C
C       STATUS-   USED TO RETURN THE I/O CHANNEL STATUS WORD.
C
C       SYSERR-   SAME AS ABOVE.
C
C       UFTBC-    NUMBER  OF BYTES IN LAST BLOCK OF  FILENAM
C                 STORED HERE.
C
C       UFTBK-    NUMBER  OF  LAST BLOCK IN  FILENAM  STORED
C                 HERE.
C
C*************************************************************

      COMPILER NOSTACK
      SUBROUTINE CHANNEL(ITASK,DIR,MODE,PCNT,DCOUNT,
     +FILENAM,DCHBLKS,DARRAY,PARRAY,ERROR,SYSERR)
```

```
C*****************************C
C       SETUP CHANNEL         C
C*****************************C

      DIMENSION PARRAY(16),IPARRAY(16),DARRAY(DCOUNT),
     +ISTAT(20),FILENAM(7)

      INTEGER CMD,PCNT,PARRAY,STATUS,ERROR,MSB,ERRBIT,
     +PCNTMSK,STATBIT,ERRMSK,DARRAY,DCOUNT,DCMD,MODE,IER,
     +ITASK,DIR,PCOUNT,ABRTERR,ISTAT,DCHBLKS,UFTBK,UFTBC,
     +FILENAM,RDOSERR,SYSERR,IPARRAY,NOABRT,EOF,IMODE

      LOGICAL BTEST

C*********************************C
C       INITILIZE VARIABLES       C
C*********************************C

      DATA ERRBIT/7/,STATBIT/6/,MSB,PCNTMSK,ERRMSK/3*15/,
     +NOABRT/255/,EOF/9/

      SYSERR=1
      ERROR=0
      ABRTERR=0
      CALL BSET(RDOSERR,MSB)

C************************************************C
C       BOUNDS CHECK FOR ARGUMENTS PASSED        C
C************************************************C

      IF(ITASK.GE.0.AND.ITASK.LE.127)GO TO 3
      ERROR=ISHFT(2,8)
      RETURN

3     IF(DCOUNT.GE.0)GO TO 5
      ERROR=ISHFT(3,8)
      RETURN

5     IF(DIR.GE.0.AND.DIR.LE.1)GO TO 8
      ERROR=ISHFT(4,8)
      RETURN

8     IF(MODE.GE.0.AND.MODE.LE.3)GO TO 10
      ERROR=ISHFT(5,8)
      RETURN

10    IF(PCNT.GE.0.AND.PCNT.LE.15)GO TO 12
      ERROR=ISHFT(6,8)
      RETURN

12    IF(DCHBLKS.GE.0)GO TO 15
      ERROR=ISHFT(8,8)
      RETURN
```

```
C*********************************************C
C       FORM COMMAND WORD, DATA COMMAND,      C
C       AND SETUP PCOUNT AND IPARRAY          C
C*********************************************C

15      CMD=ISHFT(ITASK,8).OR.ISHFT(DIR,7).OR.ISHFT(MODE,4)

C   *** SET MSB OF COMMAND WORD
        CALL BSET(CMD,MSB)

        DCMD=ISHFT(255,8).OR.CMD

        PCOUNT=PCNT
        IF(PCOUNT.EQ.0)GO TO 40
        DO 20 I=1,PCOUNT
          IPARRAY(I)=PARRAY(I)
20      CONTINUE

C*********************************C
C       SEND COMMAND WORD         C
C*********************************C

40      CALL SANDS(CMD.OR.PCOUNT,STATUS)

C   *** CHECK ERROR FLAG IN I/O CHANNEL'S RESPONSE
        IF(.NOT.BTEST(STATUS,ERRBIT))GO TO 50
        ERROR=STATUS.AND.ERRMSK.OR.ISHFT(10,8).OR.ABRTERR
        RETURN

50      IF((MODE.EQ.0).OR.(MODE.EQ.3))GO TO 100

C*************************************************************C
C       SETUP TO SEND DATA COUNT FOR MODE 1 OR MODE 2         C
C*************************************************************C

        IF(PCOUNT.EQ.0)GO TO 80

C   *** ADD DCOUNT TO BEGINNING OF PARAMETER LIST
        DO 60 I=1,PCOUNT
          IPARRAY(PCOUNT-I+2)=IPARRAY(PCOUNT-I+1)
60      CONTINUE
80      PCOUNT=PCOUNT+1
        IPARRAY(1)=DCOUNT

C*********************************************************C
C       CHECK FOR PARAMETERS AND SEND THEM IF ANY         C
C*********************************************************C

100     IF(PCOUNT.EQ.0)GO TO 300

        DO 200 I=1,PCOUNT
          CALL SANDS(IPARRAY(I),STATUS)

          IF(.NOT.BTEST(STATUS,ERRBIT))GO TO 180
```

```
C    *** PUT PARAMETER NUMBER IN PCNT FOR RETURN
           PCNT=I
           ERROR=STATUS.AND.ERRMSK.OR.ISHFT(12,8).OR.ABRTERR
           RETURN

C    *** IF ALL PARAMETERS NOT SENT BUT PCOUNT IN STATUS
C    *** IS ZERO, ABORT TASK AND RETURN AN ERROR
180        IF(.NOT.((I.NE.PCOUNT).AND.((STATUS.AND.PCNTMSK)
     +.EQ.0)))GO TO 200
           ERROR=STATUS.AND.ERRMSK.OR.ISHFT(I,4).OR.ISHFT
     +(14,8).OR.ABRTERR
           GO TO 1000

200        CONTINUE

C    *** ALL PARAMETERS SENT, CHECK FOR 0 PARAMETER
C    *** COUNT FROM I/O CHANNEL
           IF((STATUS.AND.PCNTMSK).EQ.0)GO TO 300
           ERROR=STATUS.AND.ERRMSK.OR.ISHFT(16,8).OR.ABRTERR
           GO TO 1000

C*************************************************C
C       LOOK FOR DATA TRANSFER OR DATA CHANNEL    C
C*************************************************C

C    *** OFFSET MODE BY ONE FOR COMPUTED GO TO
300        IMODE=MODE+1
           GO TO (400,2000,3000,400),IMODE

C*****************************C
C       WRAP UP COMMAND       C
C*****************************C

400        CALL CANDR(STATUS)

           IF(.NOT.BTEST(STATUS,ERRBIT))GO TO 420
           ERROR=STATUS.AND.ERRMSK.OR.ISHFT(20,8).OR.ABRTERR
           RETURN

C    *** IF S FLAG SET, RETURN
420        IF(BTEST(STATUS,STATBIT))RETURN

C    *** ELSE SEND ERROR AND ABORT CMD
           ERROR=ISHFT(22,8)

C*********************************C
C       TASK ABORT ROUTINE        C
C*********************************C

C    *** IF ABORT IS IN PROGRESS RETURN "CANNOT ABORT" ERROR
1000       IF(ABRTERR.NE.0)GO TO 1100

C    *** SET UP TO ABORT AND INITIATE THE CYCLE
           MODE=3
```

```
        CMD=0
        ABRTERR=ISHFT(1,15)
        ERROR=ABRTERR.OR.ERROR
        GO TO 12

C   *** INSERT "CANNOT ABORT" ERROR AND RETURN
1100    ERROR=ERROR.OR.ISHFT(NOABRT,8)
        RETURN

C*********************************C
C       SEND DATA ROUTINE         C
C*********************************C

2000    DO 2200 I=1,DCOUNT

C   *** SEND DATA COMMAND (DCMD)
          CALL SANDS(DCMD,STATUS)

          IF(.NOT.BTEST(STATUS,ERRBIT))GO TO 2100
C   *** RETURN DATA WORD COUNT IN DCOUNT
          DCOUNT=I
          ERROR=STATUS.AND.ERRMSK.OR.ISHFT(24,8)
          RETURN

2100      IF(DIR.EQ.1)GO TO 2150

C   *** IF INPUT GET DATA WORD
          CALL CANDR(DARRAY(I))
          GO TO 2200

C   *** ELSE SEND A DATA WORD
2150      CALL SANDS(DARRAY(I),STATUS)

          IF(.NOT.BTEST(STATUS,ERRBIT))GO TO 2200
C   *** RETURN DATA WORD COUNT IN DCOUNT
          DCOUNT=I
          ERROR=STATUS.AND.ERRMSK.OR.ISHFT(26,8)
          RETURN
2200    CONTINUE

C   *** IF DONE, WRAP IT UP
        IF(BTEST(STATUS,STATBIT))GO TO 400

C   *** ELSE ABORT & RETURN
        ERROR=ISHFT(28,8)
        GO TO 1000

C***********************************C
C       DATA CHANNEL ROUTINE        C
C***********************************C

C   *** DCOUNT MUST BE 16 OR LESS FOR DCH
3000    IF(DCOUNT.LE.16)GO TO 3020
        ERROR=ISHFT(30,8)
```

```
        GO TO 1000

3020    IF(DIR.EQ.0)GO TO 3500

C       DCH TRANSMIT
C       ************

C   *** GET STATUS OF FILENAM AND PUT IT IN ISTAT
        CALL STAT(FILENAM,ISTAT,IER)
        IF(IER.EQ.1)GO TO 3120
        SYSERR=IER
        ERROR=ISHFT(31,8)
        GO TO 1000

C   *** NUMBER OF LAST BLOCK IN FILENAM
3120    UFTBK=ISTAT(9)

C   *** NUMBER OF BYTES IN LAST BLOCK OF FILENAM
        UFTBC=ISTAT(10)

C   *** SEE IF FILENAM IS EMPTY
        IF(.NOT.((UFTBK.EQ.0).AND.(UFTBC.EQ.0)))GO TO 3130
        ERROR=ISHFT(32,8)
        GO TO 1000

C   *** MAKE SURE LAST BLOCK OF FILENAM IS FULL (512 BYTES)
3130    IF(UFTBC.EQ.512)GO TO 3140
        ERROR=ISHFT(34,8)
        GO TO 1000

C   *** IF DCOUNT IS ZERO, PUT IN DEFAULT OF 4
3140    IF(DCOUNT.NE.0)GO TO 3180
        DCOUNT=4

C   *** MAKE SURE FILENAM IS EVENLY DIVISABLE BY DCOUNT,
C   *** OFFSET UFTBK BY ONE SINCE FIRST BLOCK IS #ZERO
3180    IHOLD=MOD(UFTBK+1,DCOUNT)
        IF(IHOLD.EQ.0)GO TO 3200
        ERROR=ISHFT(35,8)
        GO TO 1000

C   *** OPEN FILENAM ON CHANNEL 4
3200    CALL OPEN(4,FILENAM,2,IER)
        IF(IER.EQ.1)GO TO 3220
        SYSERR=IER
        ERROR=ISHFT(36,8)
        GO TO 1000

C   *** SEND DATA COMMAND (DCMD)
3220    CALL SANDS(DCMD,STATUS)

        IF(.NOT.BTEST(STATUS,ERRBIT))GO TO 3240
        ERROR=STATUS.AND.ERRMSK.OR.ISHFT(38,8)
        RETURN
```

```
C   *** INITIATE DCH
3240    CALL DCHTX(DCOUNT,STATUS,RDOSERR)

        GO TO 3565

C       DCH RECEIVE
C       ***********

C   *** CREATE RANDOM FILE IF DCHBLKS IS 0
3500    IF(DCHBLKS.EQ.0)GO TO 3540

C   *** ELSE CREATE CONTIGUOUS FILE
C   *** USING DCHBLKS FOR BLOCK COUNT
        CALL CFILW(FILENAM,3,DCHBLKS,IER)
        IF(IER.EQ.1)GO TO 3560
        SYSERR=IER
        ERROR=ISHFT(40,8)
        GO TO 1000

3540    CALL CFILW(FILENAM,2,IER)
        IF(IER.EQ.1.OR.IER.EQ.12)GO TO 3560
        SYSERR=IER
        ERROR=ISHFT(42,8)
        GO TO 1000

C   *** OPEN FILENAM ON CHANNEL 4
3560    CALL OPEN(4,FILENAM,2,IER)
        IF(IER.EQ.1)GO TO 3544
        SYSERR=IER
        ERROR=ISHFT(62,8)
        GO TO 1000

C   *** SEND DATA COMMAND (DCMD)
3544    CALL SANDS(DCMD,STATUS)

        IF(.NOT.BTEST(STATUS,ERRBIT))GO TO 3564
        ERROR=STATUS.AND.ERRMSK.OR.ISHFT(46,8)
        RETURN

C   *** INITIATE DCH
3564    CALL DCHRX(DCOUNT,STATUS,RDOSERR)

C       WRAP UP DATA CHANNEL
C       ********************

C   *** MAKE RDOSERR LIKE FORTRAN SYSTEM ERRORS
3565    SYSERR=RDOSERR+3
C   *** IF MSB OF RDOSERR SET, NO ERROR SO SYSERR IS ONE
        IF(BTEST(RDOSERR,MSB))SYSERR=1

C   *** CLOSE FILE FIRST, THEN CHECK FOR ANY DCH ERRORS
        CALL CLOSE(4,IER)

        IF(.NOT.BTEST(STATUS,ERRBIT))GO TO 3575
```

```
        ERROR=STATUS.AND.ERRMSK.OR.ISHFT(52,8)
        RETURN

C   *** IGNORE "END OF FILE" SYSTEM ERROR
C   *** NOTE THAT SYSERR IS THE MODIFIED RDOSERR
3575    IF((SYSERR.EQ.1).OR.(SYSERR.EQ.EOF))GO TO 3580
        ERROR=ISHFT(54,8)
        GO TO 1000

C   *** NOW DO CLOSE FILE ERROR CHECK
3580    IF(IER.EQ.1)GO TO 3590
        SYSERR=IER
        ERROR=ISHFT(50,8)
        GO TO 1000


C   *** SEND DATA COMMAND (DCMD) WITH S FLAG SET
3590    CALL BSET(DCMD,STATBIT)
        CALL SANDS(DCMD,STATUS)

        IF(.NOT.BTEST(STATUS,ERRBIT))GO TO 3600
        ERROR=STATUS.AND.ERRMSK.OR.ISHFT(56,8)
        RETURN

C   *** IF DONE WRAP IT UP
3600    IF(BTEST(STATUS,STATBIT))GO TO 400

C   *** ELSE ABORT TASK
        ERROR=ISHFT(58,8)
        GO TO 1000

        END
```

```
;++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
;                                                                      ;
;                          NOVA/CROMEMCO                               ;
;                                                                      ;
;                       CHANNEL SUBROUTINE                             ;
;                                                                      ;
;                          S  A  N  D  S                               ;
;                                                                      ;
;                          VERSION 1.1                                 ;
;                                                                      ;
;                          - - - - - - - - - -                         ;
;                                                                      ;
;                          WRITTEN  BY                                 ;
;                                                                      ;
;                       CAPT DAN FREDAL, USAF                          ;
;                                                                      ;
;                          MARCH 1981                                  ;
;                                                                      ;
;++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
;
;       SANDS  IS  A FORTRAN  CALLABLE,  ASSEMBLY  LANGUAGE
;   SUBROUTINE.   IT  SHOULD BE ASSEMBLED WITH DG'S EXTENDED
;   ASSEMBLER.  SANDS IS CALLED AS FOLLOWS:
;
;       CALL SANDS(CMDWRD,STATUS)
;
;       CMDWRD- THIS ARGUMENT IS PASSED TO SANDS.   IT IS IN
;               TURN PLACED IN THE C PORT OF THE I/O CHANNEL.
;
;       STATUS- THIS ARGUMENT IS RETURNED BY SANDS.   IT  IS
;               THE  WORD  PLACED IN THE C PORT BY  THE  I/O
;               CHANNEL AFTER THE DONE FLAG HAS BEEN SET.
;
;       SANDS  PLACES  CMDWRD  IN  THE C PORT  OF  THE  I/O
;   CHANNEL AND SETS THE BUSY FLAG.   IT THEN WAITS FOR  THE
;   DONE  FLAG.  WHEN IT IS SET,  SANDS READS THE C PORT AND
;   RETURNS TO THE CALLING ROUTINE.
;
;**********************************************************
;
;   SET UP SANDS
        .TITL   SANDS
        .ENT    SANDS
        .EXTD   .FARL,.FRET


        .NREL

        FS.
;
```

```
;   START PROGRAM
SANDS:  JSR     @.FARL          ;REQUIRED TO SETUP FOR
                                ;FORTRAN VARIABLES

        LDA     0,@TMP,3        ;GET WORD FROM CALLING
                                ;ROUTINE
        DOCS    0,25            ;SEND IT TO I/O CHANNEL
                                ;AND SET BUSY

TEST:   SKPDN   25              ;TEST FOR DONE FLAG
        JMP     TEST            ;IF FALSE, TEST AGAIN
        DIC     0,25            ;GET I/O CHANNEL STATUS
        STA     0,@TMP+1,3      ;RETURN WORD TO CALLING ROUTINE

        JSR     @.FRET          ;RETURN TO MAIN ROUTINE

        FS.=2                   ;FRAME SIZE
        TMP=-167                ;ARGUMENT OFFSET

        .END    SANDS
```

```
;++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
;                                                                      ;
;                          NOVA/CROMEMCO                               ;
;                                                                      ;
;                        CHANNEL SUBROUTINE                            ;
;                                                                      ;
;                                                                      ;
;                            C  A  N  D  R                             ;
;                                                                      ;
;                                                                      ;
;                            VERSION 1.1                               ;
;                                                                      ;
;                          - - - - - - - - - -                         ;
;                                                                      ;
;                            WRITTEN  BY                               ;
;                                                                      ;
;                        CAPT DAN FREDAL, USAF                         ;
;                                                                      ;
;                             MARCH 1981                               ;
;                                                                      ;
;++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
;
;       CANDR  IS  A FORTRAN  CALLABLE,  ASSEMBLY  LANGUAGE
;   SUBROUTINE.  IT SHOULD BE ASSEMBLED WITH THE DG EXTENDED
;   ASSEMBLER.  CANDR IS CALLED AS FOLLOWS:
;
;       CALL CANDR(STATUS)
;
;       STATUS- THIS ARGUMENT IS RETURNED BY CANDR.   IT  IS
;               THE  WORD  PLACED IN THE C PORT BY  THE  I/O
;               CHANNEL AFTER THE DONE FLAG HAS BEEN SET.
;
;       CANDR  CLEARS  THE DONE FLAG AND WAITS FOR THE  I/O
;   CHANNEL  TO  SET IT AGAIN.   IT THEN READS THE  C  PORT,
;   CLEARS  BOTH THE BUSY AND DONE FLAGS AND RETURNS TO  THE
;   CALLING ROUTINE.
;
;**********************************************************************
;
;   SET UP CANDR
        .TITL   CANDR
        .ENT    CANDR
        .EXTD   .FARL,.FRET

        .NREL

        FS.
;
```

```
;   START PROGRAM
CANDR:  JSR     @.FARL          ;REQUIRED TO SETUP
                                ;FOR FORTRAN VARIABLES


        NIOC    25              ;CLEAR DONE FLAG

        SKPDN   25              ;TEST FOR DONE FLAG
        JMP     .-1
        DICC    0,25            ;GET STATUS FROM I/O CHANNEL
                                ;AND CLEAR BUSY AND DONE
        STA     0,@TMP,3        ;PASS IT TO CALLING ROUTINE
        JSR     @.FRET          ;RETURN TO MAIN ROUTINE

        FS.=1                   ;FRAME SIZE
        TMP=-167                ;VARIABLE OFFSET

        .END    CANDR
```

NOVA/CROMEMCO

CHANNEL SUBROUTINE

D C H T X

VERSION 1.1

- - - - - - - - - -

WRITTEN BY

CAPT DAN FREDAL, USAF

MARCH 1981

DCHTX IS A FORTRAN CALLABLE, ASSEMBLY LANGUAGE SUBROUTINE. IT SHOULD BE ASSEMBLED WITH THE DG EXTENDED ASSEMBLER. DCHTX IS CALLED AS FOLLOWS:

CALL DCHTX(DCHBLKS,STATUS,RDOSERR)

DCHBLKS- THIS ARGUMENT IS PASSED TO DCHTX. IT MUST CONTAIN THE NUMBER OF DISK BLOCKS WHICH ARE TO BE BUFFERED FOR EACH DATA CHANNEL (DCH) TRANSFER OF DATA. THIS ARGUMENT MUST NOT BE GREATER THAN 16. THE CONTENTS OF DCHBLKS IS DESTROYED BY DCHTX.

STATUS- THIS ARGUMENT IS RETURNED BY DCHTX. IT WILL BE THE WORD CONTAINED IN THE C PORT (I/O CHANNEL STATUS) WHEN DCHTX WAS EXITED.

RDOSERR- THIS ARGUMENT IS RETURNED BY DCHTX. IT WILL CONTAIN ANY RDOS SYSTEM ERROR WHICH OCCURRED AS A RESULT OF CALLING .IOPR, OR .RDB. RDOSERR REMAINS UNCHANGED IF NO SYSTEM ERROR OCCURS.

DCHTX TRANSFERS DATA FROM THE 10 MBYTE DISK TO THE I/O CHANNEL USING DCH. IT FIRST CLEARS STATUS AND THEN GETS THE RDOS EQUIVALENT OF FORTRAN CHANNEL #4. DCHTX THEN PROCEEDS TO TRANSFER DATA FROM THE DISK TO ONE OF ITS BUFFERS. AT THE SAME TIME, DCHTX HAS THE I/O CHANNEL TRANSFERRING DATA FROM ITS OTHER BUFFER. ALL TRANSFERS ARE MADE USING DCH. THIS CONTINUES UNTIL AN ERROR OCCURS OR THE I/O CHANNEL INDICATES IT IS DONE. DCHTX CHECKS THE I/O CHANNEL STATUS WORD AFTER EACH

```
;   TRANSFER  FOR THE ERROR OR DONE BIT.   IF EITHER IS  SET
;   IT RETURNS THE STATUS WORD IN STATUS AND RETURNS TO  THE
;   CALLING  ROUTINE.    IF  A  SYSTEM  ERROR  OCCURS  WHILE
;   READING  THE  DISK,  DCHTX WAITS FOR THE I/O CHANNEL  TO
;   COMPLETE ITS TRANSFER AND THEN RETURNS THE SYSTEM  ERROR
;   IN RDOSERR AND THE I/O CHANNEL'S STATUS WORD IN  STATUS.
;   IT THEN RETURNS TO THE CALLING ROUTINE.
;
;*************************************************************

;
;   SET UP DCHTX
        .TITL   DCHTX
        .ENT    DCHTX,.BUF1,.BUF2
        .EXTD   .FARL,.FRET,.IOPR

        .ZREL           ;LOCATES THE FOLLOWING ON ZERO PAGE

 .BUF1: BUFF1           ;PUT BUFFER POINTERS ON ZERO PAGE
 .BUF2: BUFF2           ;SO DCHRX.SR CAN GET AT THEM

        .NREL           ;LOCATES THE FOLLOWING NORMALLY

        FS.

;
;   CONSTANTS
THREE:  0                       ;SAVE THREE HERE

START:  0                       ;FIRST BLOCK TO BE READ,
                                ;STARTS AT BLOCK ZERO

STATMSK:177440                  ;STATUS MASK- CMD=FF,
                                ;ERROR=0,STATUS=0,MODE=2

CHANNUM:4                       ;CHANNEL THAT FILE IS
                                ;OPENED ON

BLKCNT: 0                       ;DCH BLOCK COUNT STORED
                                ;HERE, THIS IS PASSED
                                ;AS AN ARGUMENT
;
;   START ROUTINE

DCHTX:  JSR     @.FARL

;   CLEAR STATUS (TMP+1) SO ZERO IS RETURNED IF NO ERROR
        SUBO    0,0             ;CLEAR AC0
        STA     0,@TMP+1,3

;   READ FIRST CHANNEL BLOCK INTO BUFF1
        LDA     2,CHANNUM       ;PUT FORTRAN CHANNEL NUMBER
                                ;IN AC2
        STA     3,THREE         ;SAVE AC3
```

```
        JSR     @.IOPR          ;GET MATCHING RDOS CHANNEL
                                ;NUMBER
        JMP     ERROR           ;RETURN ERROR IF ANY
        LDA     3,THREE         ;RESTORE AC3

        LDA     0,@TMP,3        ;GET BLOCK COUNT
        STA     0,BLKCNT        ;SAVE BLOCK COUNT
        MOVS    0,0             ;SWAP IT INTO LEFT HALF
                                ;OF AC0
        COM     0,0             ;OR BLOCK
        AND     0,2             ;   COUNT WITH
        ADC     0,2             ;         CHANNEL NUMBER
        STA     2,@TMP,3        ;PUT THE RESULT BACK FOR
                                ;LATER REFERENCE


        LDA     0,.BUF1         ;POINT TO BUFFER #1
        LDA     1,START         ;LOGICAL BLOCK TO START
                                ;TRANSFER WITH


        .SYSTM
        .RDB    77              ;WRITE  FILE INTO BUFF1
                                ;FROM DISK
        JMP     ERROR

;   POINT TO BUFF1 AND START CHANNEL
LOOP:   LDA     0,.BUF1
        DOBC    0,25

;   SET UP TO READ NEXT CHANNEL BLOCK INTO BUFF2,
;   THEN DO IT ALL AGAIN
        LDA     2,BLKCNT        ;INCREMENT LOGICAL BLOCK
        ADD     2,1             ;POINTER BY BLKCNT
        LDA     2,@TMP,3        ;RESTORE BLOCK COUNT/CHANNEL
                                ;NUMBER WORD
        LDA     0,.BUF2         ;POINT TO BUFFER 2

;   READ NEXT CHANNEL BLOCK INTO BUFF2 AND WAIT FOR DONE
        .SYSTM
        .RDB    77              ;WRITE TO BUFF2
        JMP     ERRDONE

        LDA     0,STATMSK       ;GET READY TO CHECK
                                ;CHANNEL'S STATUS

        SKPDN   25              ;WAIT TILL I/O CHANNEL
        JMP     .-1             ;IS DONE

;   CHANNEL DONE?
        DIC     2,25            ;GET STATUS FROM CHANNEL
        SUB     2,0,SZR         ;IF NOT DONE, KEEP GOING
        JMP     RET             ;ELSE RETURN

;   POINT TO BUFF2 AND START CHANNEL
        LDA     0,.BUF2
```

```
        DOBC    0,25

;   READ NEXT CHANNEL BLOCK INTO BUFF1
        LDA     2,BLKCNT        ;INCREMENT LOGICAL BLOCK
        ADD     2,1             ;POINTER BY BLKCNT
        LDA     2,@TMP,3        ;RESTORE BLOCK COUNT/CHANNEL
                                ;NUMBER WORD
        LDA     0,.BUF1         ;POINT TO START OF BUFF1

;   READ NEXT CHANNEL BLOCK INTO BUFF1 AND WAIT FOR DONE
        .SYSTM
        .RDB    77              ;WRITE TO BUFF1
        JMP     ERRDONE

        LDA     0,STATMSK       ;GET READY TO CHECK
                                ;CHANNEL'S STATUS

        SKPDN   25              ;WAIT TILL I/O CHANNEL
        JMP     .-1             ;IS DONE

;   CHANNEL DONE?
        DIC     2,25            ;GET STATUS FROM CHANNEL
        SUB     2,0,SZR         ;IF NOT DONE, DO IT ALL AGAIN
        JMP     RET             ;ELSE RETURN

        JMP     LOOP

;
;   ERROR RETURN
ERRDONE:SKPDN   25              ;WAIT TILL I/O CHANNEL
        JMP     .-1             ;IS DONE

ERROR:  STA     2,@TMP+2,3      ;RETURN RDOS ERROR TO
                                ;CALLING ROUTINE
        DIC     2,25            ;GET CHANNEL'S STATUS
RET:    STA     2,@TMP+1,3      ;AND RETURN IT

        JSR     @.FRET          ;RETURN TO CALLING ROUTINE
;
;   STORAGE AREA
BUFF1:  .BLK    4096.           ;TWO DATA BUFFERS FOR
BUFF2:  .BLK    4096.           ;TRANSFER OF DATA TO
                                ;OR FROM I/O CHANNEL.  EACH
                                ;BUFFER HOLDS UP TO 16
                                ;BLOCKS OF DATA


        FS.=3                   ;FRAME SIZE
        TMP=-167                ;VARIABLE OFFSET

        .END    DCHTX
```

NOVA/CROMEMCO

CHANNEL SUBROUTINE

D C H R X

VERSION 1.1

- - - - - - - - - - -

WRITTEN BY

CAPT DAN FREDAL, USAF

MARCH 1981

DCHRX IS A FORTRAN CALLABLE, ASSEMBLY LANGUAGE SUBROUTINE. IT SHOULD BE ASSEMBLED WITH THE DG EXTENDED ASSEMBLER. DCHRX IS CALLED AS FOLLOWS:

CALL DCHRX(DCHBLKS,STATUS,RDOSERR)

DCHBLKS- THIS ARGUMENT IS PASSED TO DCHRX. IT MUST CONTAIN THE NUMBER OF DISK BLOCKS WHICH ARE TO BE BUFFERED FOR EACH DATA CHANNEL (DCH) TRANSFER OF DATA. THIS ARGUMENT MUST NOT BE GREATER THAN 16. THE CONTENTS OF DCHBLKS IS DESTROYED BY DCHRX.

STATUS- THIS ARGUMENT IS RETURNED BY DCHRX. IT WILL BE THE WORD CONTAINED IN THE C PORT (I/O CHANNEL STATUS) WHEN DCHRX WAS EXITED.

RDOSERR- THIS ARGUMENT IS RETURNED BY DCHRX. IT WILL CONTAIN ANY RDOS SYSTEM ERROR WHICH OCCURRED AS A RESULT OF CALLING .IOPR, OR .WRB. RDOSERR REMAINS UNCHANGED IF NO SYSTEM ERROR OCCURS.

DCHRX TRANSFERS DATA FROM THE I/O CHANNEL TO THE 10 MBYTE DISK USING DCH. IT FIRST CLEARS STATUS AND THEN GETS THE RDOS EQUIVALENT OF FORTRAN CHANNEL #4. DCHRX THEN PROCEEDS TO TRANSFER DATA FROM THE I/O CHANNEL TO ONE OF ITS BUFFERS. AT THE SAME TIME, DCHRX WRITES THE CONTENTS OF ITS OTHER BUFFER TO A FILE ON THE DISK. ALL TRANSFERS ARE MADE USING DCH. THIS CONTINUES UNTIL AN ERROR OCCURS OR THE I/O CHANNEL INDICATES IT IS DONE. DCHRX CHECKS THE I/O CHANNEL STATUS WORD AFTER EACH

```
;   TRANSFER  FOR THE ERROR OR DONE BIT.   IF EITHER IS  SET
;   IT   WRITES THE REMAINING BUFFER TO THE DISK AND RETURNS
;   THE  STATUS WORD IN STATUS AND RETURNS TO   THE  CALLING
;   ROUTINE.   IF  A SYSTEM ERROR OCCURS WHILE  WRITING  THE
;   DISK,  DCHRX  WAITS FOR THE I/O CHANNEL TO  COMPLETE ITS
;   TRANSFER  AND THEN RETURNS THE SYSTEM  ERROR IN  RDOSERR
;   AND  THE I/O CHANNEL'S STATUS WORD IN  STATUS.  IT  THEN
;   RETURNS TO THE CALLING ROUTINE.
;
;*****************************************************************
;
;
;   REQUIRED SYSTEM CALLS
;
        .TITL   DCHRX
        .ENT    DCHRX
        .EXTD   .FARL,.FRET,.IOPR,.BUF1,.BUF2
        .NREL

        FS.

;
;   CONSTANTS
THREE:  0                       ;SAVE AC3 HERE

START:  0                       ;FIRST BLOCK TO BE READ,
                                ;STARTS AT BLOCK ZERO

STATMSK:177440                  ;STATUS MASK- CMD=FF,
                                ;ERROR=0,STATUS=0,MODE=2

CHANNUM:4                       ;CHANNEL THAT FILE IS
                                ;OPENED ON

BLKCNT: 0                       ;DCH BLOCK COUNT STORED
                                ;HERE, THIS IS PASSED
                                ;AS AN ARGUMENT
;
;   START ROUTINE

DCHRX:  JSR     @.FARL

;   CLEAR STATUS (TMP+1) SO ZERO IS RETURNED IF NO ERROR
        SUBO    0,0             ;CLEAR AC0
        STA     0,@TMP+1,3

;   GET RDOS CHANNEL NUMBER
        LDA     2,CHANNUM       ;PUT FORTRAN CHANNEL NUMBER
                                ;IN AC2
        STA     3,THREE         ;SAVE AC3
        JSR     @.IOPR          ;GET MATCHING RDOS CHANNEL
                                ;NUMBER
        JMP     ERROR           ;RETURN ERROR IF ANY
        LDA     3,THREE         ;RESTORE AC3
```

```
;
;   GET AND STORE CH BLOCK COUNT,
;   CREATE BLOCK COUNT/CHANNEL WORD FOR AC2
        LDA     0,@TMP,3        ;GET BLOCK COUNT
        STA     0,BLKCNT        ;SAVE BLOCK COUNT
        MOVS    0,0             ;SWAP IT INTO LEFT HALF
                                ;OF AC0
        COM     0,0             ;OR BLOCK
        AND     0,2             ;   COUNT WITH
        ADC     0,2             ;         CHANNEL NUMBER
        STA     2,@TMP,3        ;PUT THE RESULT BACK FOR
                                ;LATER REFERENCE

;   SETUP AC1 SO BLOCK POINTER WILL START AT ZERO
;   WHEN BLKCNT IS ADDED TO IT
        LDA     1,BLKCNT        ;PUT BLOCK COUNT IN AC1
        NEG     1,1             ;AND NEGATE IT

;   POINT TO BUFF1 AND START CHANNEL
        LDA     0,.BUF1
        DOBC    0,25

LOOP:   SKPDN   25              ;WAIT FOR DONE
        JMP     .-1

;   CHANNEL DONE?
        DIC     2,25            ;GET COMMAND FROM CHANNEL
        LDA     0,STATMSK       ;CHECK ITS STATUS
        SUB#    2,0,SZR         ;IF NOT DONE GO ON
        JMP     B2RET           ;ELSE RETURN

;   POINT TO BUFF2 AND START CHANNEL
        LDA     0,.BUF2
        DOBC    0,25

;   WRITE BUFF1 ONTO THE DISK
        LDA     2,BLKCNT        ;INCREMENT LOGICAL BLOCK
        ADD     2,1             ;POINTER BY BLKCNT
        LDA     2,@TMP,3        ;RESTORE BLOCK COUNT/CHANNEL
                                ;NUMBER WORD
        LDA     0,.BUF1         ;POINT TO START OF BUFF1

        .SYSTM
        .WRB    77              ;WRITE TO FILE ON DISK
                                ;FROM BUFF1
        JMP     ERRDONE

        SKPDN   25              ;WAIT FOR DONE
        JMP     .-1

;   CHANNEL DONE?
        DIC     2,25            ;GET COMMAND FROM CHANNEL
        LDA     0,STATMSK       ;CHECK ITS STATUS
        SUB     2,0,SZR         ;IF NOT DONE GO ON
```

```
        JMP     B1RET           ;ELSE RETURN

;   POINT TO BUFF1 AND START CHANNEL
        LDA     0,.BUF1
        DOBC    0,25

;   WRITE BUFF2 ONTO THE DISK AND WAIT FOR DONE
        LDA     2,BLKCNT        ;INCREMENT LOGICAL BLOCK
        ADD     2,1             ;POINTER BY BLKCNT
        LDA     2,@TMP,3        ;RESTORE BLOCK COUNT/CHANNEL
                                ;NUMBER WORD
        LDA     0,.BUF2         ;POINT TO START OF BUFF2

        .SYSTM
        .WRB    77              ;WRITE TO FILE ON DISK
                                ;FROM BUFF2
        JMP     ERRDONE

        JMP     LOOP

;   WRITE BUFF1 ONTO THE DISK AND RETURN
B1RET:  LDA     0,.BUF1         ;POINT TO START OF BUFF1
        JMP     BRET            ;AND CONTINUE

;   WRITE BUFF2 ONTO THE DISK AND RETURN
B2RET:  LDA     0,.BUF2         ;POINT TO START OF BUFF2

BRET:   STA     2,@TMP+1,3      ;RETURN CHANNEL ERROR TO
                                ;CALLING ROUTINE
        LDA     2,BLKCNT        ;INCREMENT LOGICAL BLOCK
        ADD     2,1             ;POINTER BY BLKCNT
        LDA     2,@TMP,3        ;RESTORE BLOCK COUNT/CHANNEL
                                ;NUMBER WORD

        .SYSTM
        .WRB    77              ;WRITE TO FILE ON DISK
        JSR     @.FRET          ;RETURN TO CALLING ROUTINE

;
;   ERROR RETURN
ERRDONE:SKPDN   25              ;WAIT FOR DONE
        JMP     .-1

ERROR:  STA     2,@TMP+2,3      ;RETURN RDOS ERROR TO
                                ;CALLING ROUTINE
        DIC     2,25            ;GET AND RETURN CHANNEL'S
        STA     2,@TMP+1,3      ;STATUS
        JSR     @.FRET          ;RETURN TO CALLING ROUTINE

        FS.=3
        TMP=-167

        .END    DCHRX
```

Appendix E

I/O Channel Communications Protocol

The following pages provide vertical time-line diagrams of the transactions that take place over the I/O Channel data path during the execution of each of the possible mode commands. It should be noted that, in order to provide insight into the communication protocol utilized by the I/O Channel during command and data transfers, these diagrams illustrate only the error-free transfer case. The response to error conditions occurs immediately after the error is detected. If all possible combinations of these possibilities were shown, the diagrams would become extremely cumbersome. For further information on the error possibilities, the main body of this document and the appropriate software listings should be consulted.

The quantities enclosed in quotation marks indicate explanatory comments and do not necessarily represent the actual information words that are passed over the data path. The quantities enclosed in parenthesis indicate actions which cause the Nova BUSY and DONE flags to be either set or cleared.

(SB) = Set Nova BUSY flag.
(SD) = Set Nova DONE flag.
(CD) = Clear Nova DONE flag.
(SBCD) = Set Nova BUSY and clear Nova DONE flags.
(SDCB) = Set Nova DONE and clear Nova BUSY flags.

The remaining abbreviations in the diagrams are used to depict the command word fields and the values to be loaded into these fields.

D = Command Direction/Error field.
S = Command Status field.
P = Command Parameter Count/Error Code field.

TP = Total parameter count.
RP = Remaining parameter count.
1 = Set condition.
0 = Cleared condition.

MODE "00" COMMANDS - CROMEMCO TO PERIPHERAL TASK

NOVA	CROMEMCO
Task Command with D=1 (SB)	
	Echo Task Command with P=TP (SDCB)
First Parameter (SBCD)	
	Echo Task Command with P=RP (SDCB)
Next Parameter (SBCD)	
	Echo Task Command with P=RP (SDCB)
"Continue until next to last parameter is sent"	
Last Parameter (SBCD)	
	Echo Task Command with P=0 (SDCB)
(CD)	
	"Execute task"
	Echo Task Command with S=1 (SD)
(CD)	
	"Await Next Command"

MODE "00" COMMANDS - PERIPHERAL TO CROMEMCO TASK

NOVA	CROMEMCO
Task Command with D=0 (SB)	
	Echo Task Command with P=TP (SDCB)
First Parameter (SBCD)	
	Echo Task Command with P=RP (SDCB)
Next Parameter (SBCD)	
	Echo Task Command with P=RP (SDCB)
"Continue until next to last parameter is sent"	
Last Parameter (SBCD)	
	Echo Task Command with P=0 (SDCB)
(CD)	
	"Execute task"
	Echo Task Command with S=1 (SD)
(CD)	
	"Await Next Command"

MODE "01" COMMANDS - NOVA TO CROMEMCO TRANSFER

NOVA	CROMEMCO
Task Command with D=1 (SB)	
	Echo Task Command with P=TP (SDCB)
Data Word Count Parameter (SBCD)	
	Echo Task Command with P=TP (SDCB)
Next Parameter (SBCD)	
	Echo Task Command with P=RP (SDCB)
"Continue until next to last parameter is sent"	
Last Parameter (SBCD)	
	Echo Task Command with P=0 (SDCB)
Data Command with D=1 (SBCD)	
	Echo Data Command (SDCB)
First Data Word (SBCD)	
	Echo Data Command (SDCB)
Data Command with D=1 (SBCD)	
	Echo Data Command (SDCB)
Next Data Word (SBCD)	
	Echo Data Command (SDCB)
"Continue until the next to last data word is sent"	
Data Command with D=1 (SBCD)	
	Echo Data Command (SDCB)
Last Data Word (SBCD)	
	Echo Data Command with S=1 (SDCB)
(CD)	
	"Execute Task"
	Echo Task Command with S=1 (SD)
(CD)	
	"Await Next Command"

MODE "01" COMMANDS - CROMEMCO TO NOVA TRANSFER

NOVA	CROMEMCO
Task Command with D=0 (SB)	
	Echo Task Command with P=TP (SDCB)
Data Word Count Parameter (SBCD)	
	Echo Task Command with P=TP (SDCB)
Next Parameter (SBCD)	
	Echo Task Command with P=RP (SDCB)
"Continue until next to last parameter is sent"	
Last Parameter (SBCD)	
	Echo Task Command with P=0 (SDCB)
Data Command with D=0 (SBCD)	"Execute task"
	Echo Data Command (SDCB)
(CD)	
	First Data Word (SD)
Data Command with D=0 (SBCD)	
	Echo Data Command (SDCB)
(CD)	
	Next Data Word (SD)
Data Command with D=0 (SBCD)	
	Echo Data Command (SDCB)
"Continue until the next to last data word is sent"	
Data Command with D=0 (SBCD)	
	Echo Data Command with S=1 (SDCB)
(CD)	
	Last Data Word (SD)
(CD)	
	Echo Task Command with S=1 (SD)
(CD)	
	"Await Next Command"

MODE "10" COMMANDS - NOVA TO CROMEMCO TRANSFER

NOVA	CROMEMCO
Task Command with D=1 (SB)	
	Echo Task Command with T=TP (SDCB)
DCH Block Count Parameter (SBCD)	
	Echo Task Command with P=TP (SDCB)
Next Parameter (SBCD)	
	Echo Task Command with P=RP (SDCB)
"Continue until next to last parameter is sent"	
Last Parameter (SBCD)	
	Echo Task Command with P=0 (SDCB)
Data Command with D=1 (SBCD)	
	Echo Data Command (SDCB)
(CD) First Data Block	
	Echo Data Command (SD)
(CD) Next Data Block	
	Echo Data Command (SD)
"Continue until the last data word is sent"	
Data Command with D=1 and S=1 (SBCD)	
	Echo Data Command with S=1 (SDCB)
(CD)	
	"Execute Task"
	Echo Task Command with S=1 (SD)
(CD)	
	"Await Next Command"

MODE "10" COMMANDS - CROMEMCO TO NOVA TRANSFER

NOVA	CROMEMCO
Task Command with D=0 (SB)	
	Echo Task Command with P=TP (SDCB)
DCH Block Count Parameter (SBCD)	
	Echo Task Command with P=TP (SDCB)
Next Parameter (SBCD)	
	Echo Task Command with P=RP (SDCB)
"Continue until next to last parameter is sent"	
Last Parameter (SBCD)	
	Echo Task Command with P=0 (SDCB)
Data Command with D=0 (SBCD)	"Execute task"
	Echo Data Command (SDCB)
(CD)	
	First Data Block Echo Data Command (SD)
(CD)	
	Next Data Block Echo Data Command (SD)
"Continue until the last data word is sent"	
Data Command with D=1 and S=1 (SBCD)	
	Echo Data Command with S=1 (SDCB)
(CD)	
	Echo Task Command with S=1 (SD)
(CD)	
	"Await Next Command"

MODE "11" COMMANDS - CROMEMCO TO PERIPHERAL TASK

NOVA	CROMEMCO
Task Command with D=1 (SB)	
	Echo Task Command with P=TP (SDCB)
First Parameter (SBCD)	
	Echo Task Command with P=RP (SDCB)
Next Parameter (SBCD)	
	Echo Task Command with P=RP (SDCB)

"Continue until next to last parameter is sent"

Last Parameter (SBCD)	
	Echo Task Command with P=0 (SDCB)
(CD)	"Abort task"
	Echo Task Command with S=1 (SD)
(CD)	
	"Await Next Command"

MODE "11" COMMANDS - PERIPHERAL TO CROMEMCO TASK

NOVA	CROMEMCO
Task Command with D=0 (SB)	
	Echo Task Command with P=TP (SDCB)
First Parameter (SBCD)	
	Echo Task Command with P=RP (SDCB)
Next Parameter (SBCD)	
	Echo Task Command with P=RP (SDCB)
"Continue until next to last parameter is sent"	
Last Parameter (SBCD)	
	Echo Task Command with P=0 (SDCB)
(CD)	"Abort task"
	Echo Task Command with S=1 (SD)
(CD)	
	"Await Next Command"

Appendix F

I/O Channel Error Codes

The CHANNEL subroutine returns two variables (ERROR and SYSERR) that indicate errors which occur during I/O Channel operations. SYSERR will contain any RDOS errors generated during a call to a system subroutine from either Fortran or assembly language. A value of one is returned in SYSERR if no RDOS errors occur. All RDOS errors returned by Fortran are offset by a factor of +3. CHANNEL also offsets RDOS errors returned from its assembly language routines by a factor of +3 to maintain consistency with the FORTRAN errors. The appropriate DG manual should be referenced for an explanation of the RDOS system errors.

The error information returned by ERROR is generated from error checks made within CHANNEL and the CHOPS. ERROR contains two fields. Its least significant byte contains the error that the CHOPS returns while its most significant byte contains the error returned by CHANNEL. In addition, if ERROR is negative or its most significant bit is set, a command abort was initiated by CHANNEL. When CHANNEL initiates an abort, the CHOPS error indicated in ERROR, if any, will be an error that occurred during the execution of the abort command.

The following list has the decimal value of the error code that will be returned in ERROR's most significant byte.

ERROR CODE	DESCRIPTION
0	No error occurred.
2	The TASK argument passed to CHANNEL was out of bounds (TASK<0 or TASK>127).
3	The DCOUNT argument passed to CHANNEL was out of bounds (DCOUNT<0).
4	The DIR argument passed to CHANNEL was out of bounds (DIR<0 or DIR>1).
5	The MODE argument passed to CHANNEL was out of bounds (MODE<0 or MODE>3).
6	The PCNT argument passed to CHANNEL was out of bounds (PCNT<0 or PCNT>15).
8	The DCHBLKS argument passed to channel was out of bounds (DCHBLKS<0).
10	The CHOPS returned an error after receiving the command word.
12	The CHOPS returned an error after receiving a parameter. The number of the parameter which caused the error is returned in PCNT.
14	The CHOPS counted down the parameters faster than CHANNEL. The CHANNEL parameter number where this occurred is passed in PCNT. Task abort initiated.
16	The CHOPS did not finish counting the parameters when CHANNEL did. Task abort initiated.
20	The CHOPS returned an error during the execution of a task.
22	The CHOPS did not properly terminate the task. Task abort initiated.
24	Data Transfer error. The CHOPS returned an error after receiving a Data Next command. The number of the data word is returned in DCOUNT.
26	Data Transfer error. The CHOPS returned an error after receiving a data word. The number of the data word is returned in DCOUNT.

28	Data Transfer error. Data Transfer not terminated properly by the CHOPS. Task abort initiated.
30	DCH Transfer error. DCOUNT out of bounds (DCOUNT>16).
31	DCH output error. System error returned by subroutine STAT. Task abort initiated.
32	DCH output error. File to be transferred is empty. Task abort initiated.
34	DCH output error. File to be transferred does not fill last block. Task abort initiated.
35	DCH output error. File to be transferred not evenly divisible by channel block count. Task abort initiated.
36	DCH output error. System error returned by subroutine OPEN. Task abort initiated.
38	DCH output error. Error returned by the CHOPS after receiving the Data Command initiating a DCH transfer.
40	DCH input error. System error returned by subroutine CFILW (create contiguous file). Task abort initiated.
42	DCH input error. System error returned by subroutine CFILW (create random file). Task abort initiated.
44	DCH input error. System error returned by subroutine OPEN. Task abort initiated.
46	DCH input error. Error returned by the CHOPS after Data Command sent.
50	DCH transfer error. System error returned by subroutine CLOSE. Task abort initiated.
52	DCH transfer error. Error returned by the CHOPS during DCH. System error may also be returned.
[illegible]	DCH transfer error. System error returned during DCH. This error may be generated by subroutine IOPR or by RDOS system calls .RDB (DCH output) or .WRB (DCH input). Task abort is initiated as a result of this error. Note

that CHANNEL ignores EOF system error (Fortran error #9); however, this error code will be returned in SYSERR. In addition, if no error is returned, SYSERR is set to one.

56 DCH transfer error. Error returned by the CHOPS after Data Command was sent to terminate the transfer.

58 DCH transfer error. DCH transfer not terminated properly. Task abort initiated.

255 Unable to abort task in I/O Channel. User must depress reset on the Cromemco.

The following list contains the decimal value of the CHOPS error code. These codes are returned as the least significant byte of the ERROR variable.

ERROR CODE	DESCRIPTION
0	No error occurred.
1	Not a command. This error indicates that the CHOPS was waiting for a command, but it received a word from the Nova that did not have the most significant bit set.
2	Invalid command. This error indicates that a valid command word was received, but that the task that this command is attempting to invoke has not yet been implemented.
3	Invalid parameter count. This error code indicates that the requested task requires a different number of parameters than the number of parameters indicated in the parameter count of the command word.
4	Parameter expected. This error code indicates that the CHOPS was waiting for a parameter word which has its most significant bit cleared to be sent, but the word that was received had the most significant bit set.
5	Parameter out of range. This error code indicates that the received parameter is either less than or equal to the preset low

bound or it is greater than the preset high bound for this task.

6 Invalid command mode. This error code indicates that the requested mode for this command is not implemented by the task.

7 Data command expected. This error code indicates that the CHOPS was in a data collection routine when a command was received that it did not recognize. When this occurs the data transmission is no longer properly synchronized.

8 Data buffer size exceeded. This error code indicates that the data buffer allocated by the task has overflowed and that the excess data has probably been lost.

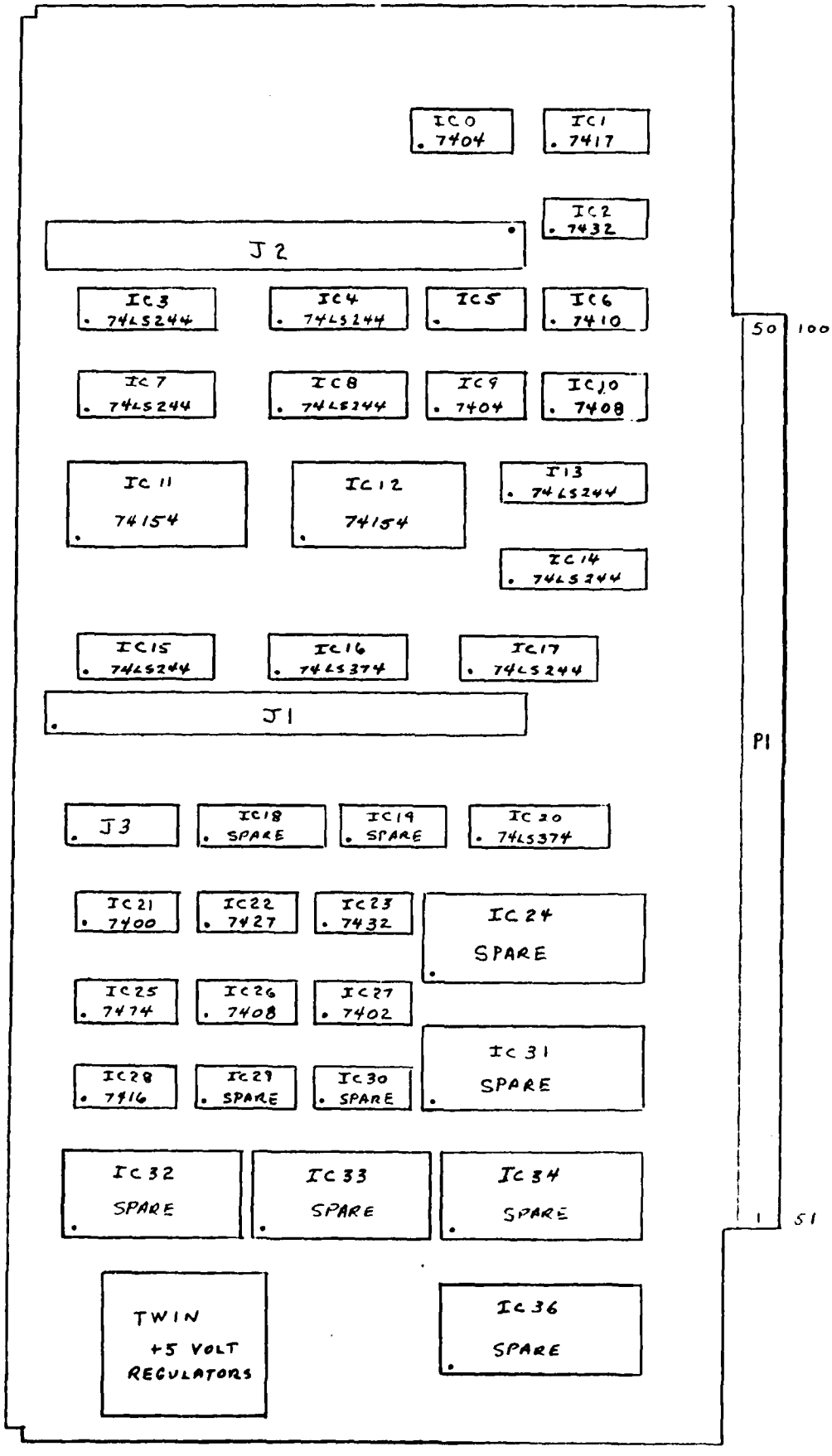
9 Data out of range. This error code indicates that the data that was transmitted or collected is either larger or smaller than the values expected for this task.

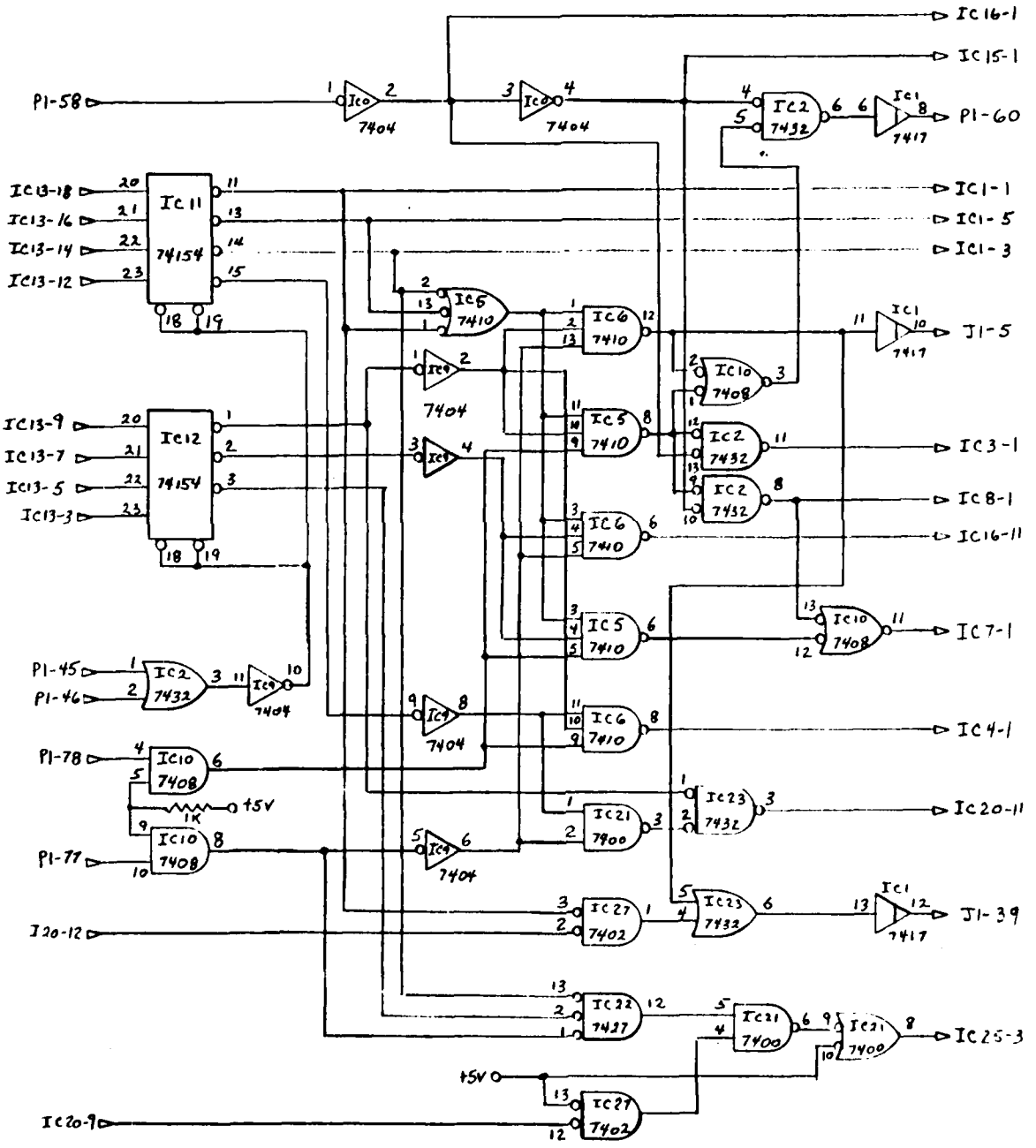
10 Requested device off-line. This error code indicates that the task requires the use of hardware that is not installed or that does not have power applied.

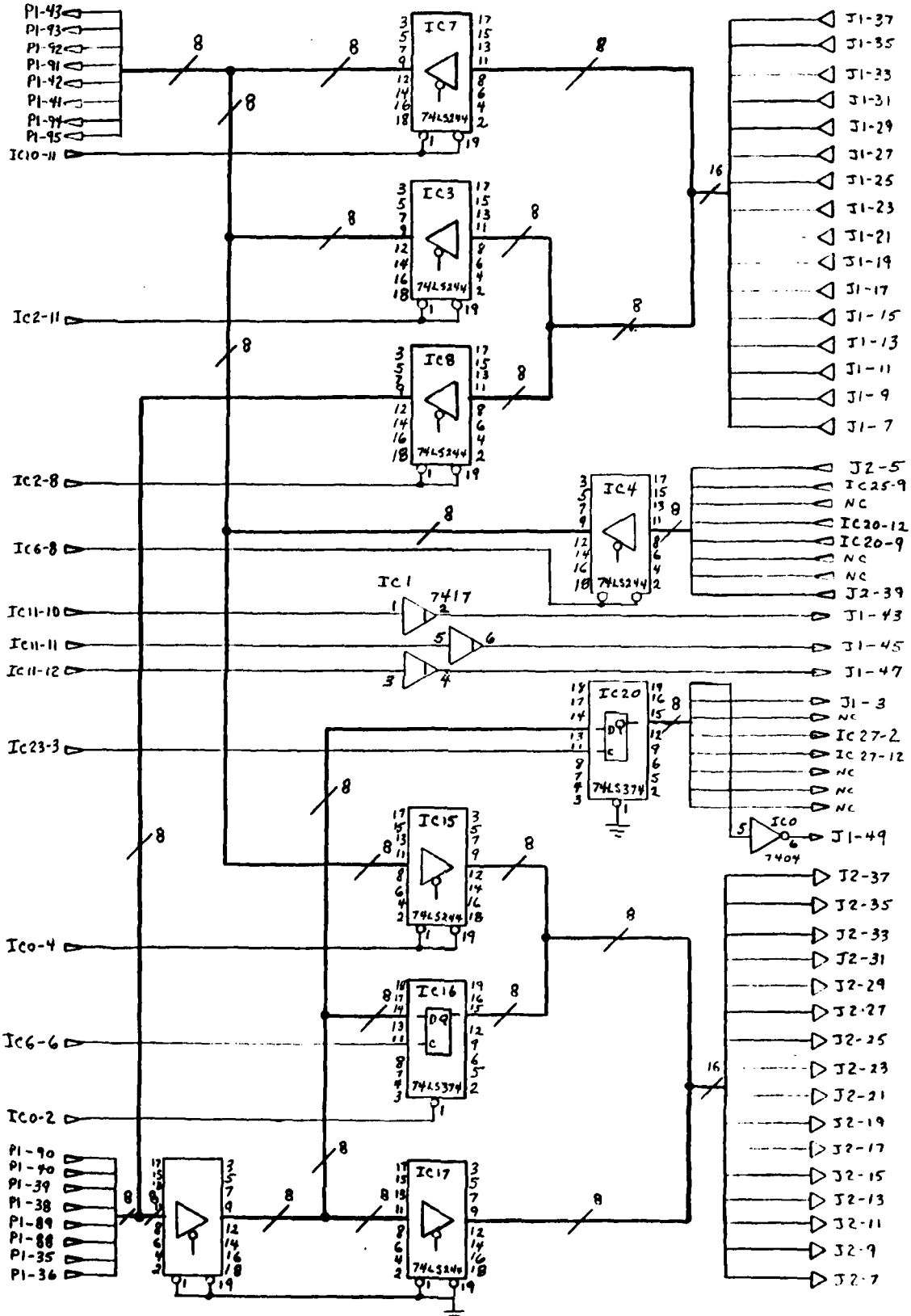
11-15 Not yet defined.

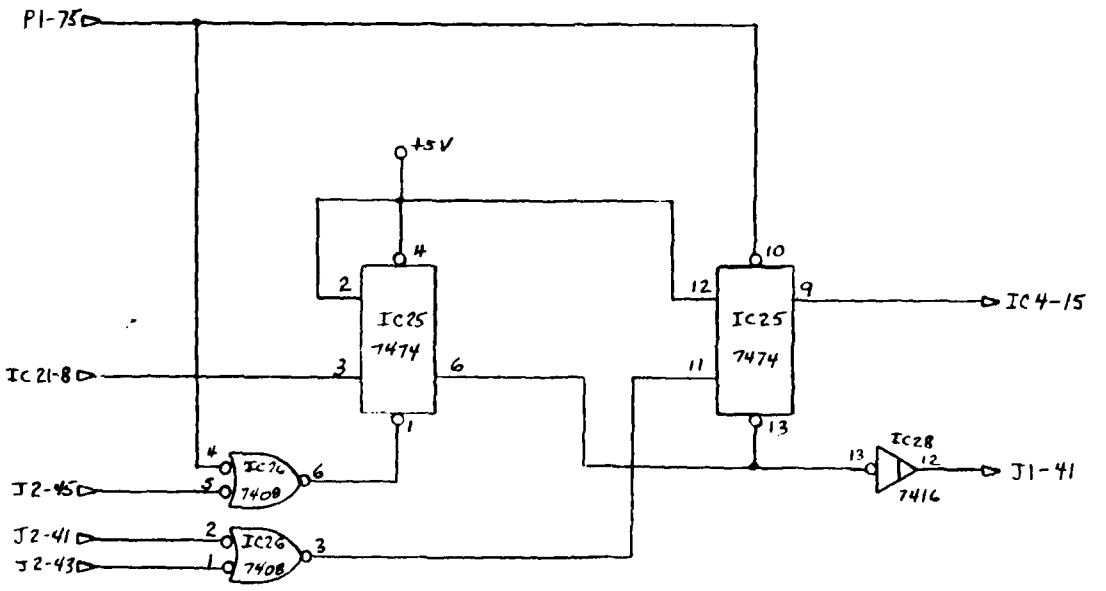
Appendix G

Cromemco Interface Schematic


IC0 7404
IC1 7417
IC2 7432
J2
IC3 74LS244
IC4 74LS244
IC5
IC6 7410
IC7 74LS244
IC8 74LS244
IC9 7404
IC10 7408
IC 11 74154
IC 12 74154
I13 74LS244
IC14 74LS244
IC15 74LS244
IC16 74LS374
IC17 74LS244
J1
J3
IC18 SPARE
IC19 SPARE
IC 20 74LS374
IC21 7400
IC22 7427
IC23 7432
IC 24 SPARE
IC25 7474
IC26 7408
IC27 7402
IC 31 SPARE
IC28 7416
IC29 SPARE
IC30 SPARE
IC 32 SPARE
IC 33 SPARE
IC 34 SPARE
TWIN +5 VOLT REGULATORS
IC 36 SPARE
50
100
P1
1
S1


P1-58
IC0
7404
IC4
7404
IC16-1
IC15-1
IC2
7432
IC1
7417
P1-60
IC13-18
IC13-16
IC13-14
IC13-12
IC11
74154
IC1-1
IC1-5
IC1-3
IC5
7410
IC6
7410
IC1
7417
J1-5
IC10
7408
IC13-9
IC13-7
IC13-5
IC13-3
IC12
74154
IC5
7410
IC2
7432
IC3-1
IC2
7432
IC8-1
IC6
7410
IC16-11
IC10
7408
IC7-1
P1-45
P1-46
IC2
7432
IC9
7404
IC5
7410
P1-78
IC10
7408
+5V
1K
IC6
7410
IC4-1
IC23
7432
IC20-11
P1-77
IC10
7408
IC21
7400
IC20-12
IC27
7402
IC23
7432
IC1
7417
J1-39
IC22
7427
IC21
7400
IC21
7400
IC25-3
+5V
IC20-9
IC27
7402


P1-43
P1-93
P1-92
P1-91
P1-42
P1-41
P1-94
P1-95
IC10-11
IC7
74LS244
IC3
74LS244
IC8
74LS244
Ic2-11
Ic2-8
Ic6-8
IC4
74LS244
IC1
7417
Ic11-10
Ic11-11
Ic11-12
Ic23-3
IC20
74LS374
IC0
7404
IC15
74LS244
IC16
74LS374
Ico-4
Ic6-6
Ico-2
P1-90
P1-90
P1-39
P1-38
P1-89
P1-88
P1-35
P1-36
IC17
74LS240
J1-37
J1-35
J1-33
J1-31
J1-29
J1-27
J1-25
J1-23
J1-21
J1-19
J1-17
J1-15
J1-13
J1-11
J1-9
J1-7
J2-5
IC25-9
NC
IC20-12
IC20-9
NC
NC
J2-39
J1-43
J1-45
J1-47
J1-3
NC
IC27-2
IC27-12
NC
NC
NC
J1-49
J2-37
J2-35
J2-33
J2-31
J2-29
J2-27
J2-25
J2-23
J2-21
J2-19
J2-17
J2-15
J2-13
J2-11
J2-9
J2-7


P1-75
+5V
IC25
7474
IC25
7474
IC 4-15
Ic 21-8
IC26
7408
J2-45
J2-41
J2-43
IC26
7408
IC28
7416
J1-41

Appendix H

I/O Channel Data Path Interconnections

Nova GPI W/W Pin	Nova Paddle Conn. Pin	Cromemco Jack/Pin	Signal Name
NC	1	1/01	Ground**
5	2	1/03	M1 - DCH control
70	3	1/05	CLOCK DATA
2	4	1/07	Nova input bit-0
6	5	1/09	Nova input bit-1
1	6	1/11	Nova input bit-2
49A	7	1/13	Nova input bit-3
49	8	1/15	Nova input bit-4
48	9	1/17	Nova input bit-5
6A	10	1/19	Nova input bit-6
41	11	1/21	Nova input bit-7
7	12	1/23	Nova input bit-8
29	13	1/25	Nova input bit-9
34	14	1/27	Nova input bit-10
8	15	1/29	Nova input bit-11
8A	16	1/31	Nova input bit-12
10	17	1/33	Nova input bit-13
9	18	1/35	Nova input bit-14
57	19	1/37	Nova input bit-15
67	20	1/39	DEVICE COMPLETE#
2A	21	1/41	DCHREQ
68A	22	1/43	A SELECT#
3	23	1/45	B SELECT#
68	24	1/47	C SELECT#
4	25	1/49	M0 - DCH OUT#/IN
69	26	2/01	no connection
70A	27	2/03	no connection
4A	28	2/05	DONE
71	29	2/07	Nova output bit-0
72A	30	2/09	Nova output bit-1
72	31	2/11	Nova output bit-2
76	32	2/13	Nova output bit-3
83	33	2/15	Nova output bit-4
84A	34	2/17	Nova output bit-5
89	35	2/19	Nova output bit-6
93	36	2/21	Nova output bit-7
82	37	2/23	Nova output bit-8
131	38	2/25	Nova output bit-9
132A	39	2/27	Nova output bit-10
132	40	2/29	Nova output bit-11
133	41	2/31	Nova output bit-12
134A	42	2/33	Nova output bit-13
98A	43	2/35	Nova output bit-14
104	44	2/37	Nova output bit-15
134	45	2/39	BUSY
135	46	2/41	DCHSEL*DCHO#
136A	47	2/43	DCHSEL*DCHI#
125	48	2/45	DCHACK#
136	49	2/47	no connection
NC	50	2/49	Nova +5 volts

**All Cromemco interface even numbered pins are grounded.

VITA

Dan Fredal was born on 31 December 1948 in Copenhagen, Denmark. He became a naturalized citizen of the United States in San Diego, California in May 1960. He graduated from high school in Imperial Beach, California in 1966 and attended San Diego State University. He enlisted in the USAF on July 10, 1968 and entered the aircraft refueling career field. In 1971 he crossed-trained into Air Traffic Control Radar Maintenance. While stationed at Nellis AFB, Nevada he attended the University of Nevada at Las Vegas from which he received the degrees of Bachelor of Science in Engineering and Bachelor of Science (Applied Physics) in May 1976. Upon completion of Officers Training School in October 1976, he received a commission in the USAF. He then served as a Program Analyst for the 6510 Test Wing at the Air Force Flight Test Center, Edwards AFB, California. He was responsible for programs such as the B-1 flight test program, the KC-10 flight test program, AF support for the Space Shuttle, and the B-52 Offensive Avionics program. In June 1979, he entered the School of Engineering, Air Force Institute of Technology.

Permanent Address: 3815 Nellis Blvd.
LasVegas, Nevada 89110

George Carroll Beasley, Jr. was born on 3 October 1946 in Newport News, Virginia. He graduated from Fair Park High School in Shreveport, Louisiana in 1964. In April of 1966 he entered the USAF. He received electronics training at Lowry AFB, Colorado, and in January of 1967 was assigned to the 1155th Tech Ops Squadron, McClellan AFB, California as a depot level electronics maintenance technician. He attended evening classes at American River College in Sacramento, California, where he received his Associate of Arts in Mathematics and Physical Science in June of 1973. In August of 1973 he was accepted into the Airman Education and Commissioning Program and was sent to Louisiana Tech University in Ruston, Louisiana, where in November of 1975 he received his Bachelor of Science in Electrical Engineering as a Cum Laude graduate. In March of 1976, he became an Honor Graduate of the Officers Training School at Lackland AFB, Texas and was commissioned a Second Lieutenant. He was then assigned to the Air Force Armament Lab at Eglin AFB, Florida, as a Digital Electronics Engineer. While there, he became the Armament Lab's 1979 Engineer of the Year and was awarded the Air Force Meritorious Service Medal. In August 1979, he entered the Air Force Institute of Technology School of Engineering at Wright Patterson AFB, Ohio. He is a member of the Tau Beta Pi, Eta Kappa Nu, and Phi Kappa Phi honor societies.

Permanent address: 3259 Penick Street
Shreveport, Louisiana 71109

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER AFIT/GE/EE/81M-2	2. GOVT ACCESSION NO. AD-A103	3. RECIPIENT'S CATALOG NUMBER 398
4. TITLE (and Subtitle) AN ANALOG SPEECH I/O CHANNEL FOR THE NOVA 2 COMPUTER BASED ON THE S-100 BUS		5. TYPE OF REPORT & PERIOD COVERED MS Thesis 6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) George C. Beasley, Jr., Capt., USAF Dan Fredal, Capt., USAF		8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS Air Force Institute of Technology (AFIT-EN) Wright-Patterson AFB, Ohio 45433		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Electrical Engineering Dept. Air Force Institute of Technology Wright Patterson AFB, Ohio 45433		12. REPORT DATE March, 1981 13. NUMBER OF PAGES
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)		15. SECURITY CLASS. (of this report) 15a. DECLASSIFICATION DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

Approved for public release; IAW AFR 190-17
Fredric C. Lynch, Major, USAF
Director of Public Affairs
21 AUG 1981

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Microprocessor
Data Acquisition
Speech Processing
Video Processing
I/O Channel
Interprocessor Interface
Operating System

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

The requirements for a capability to provide easy sampling, digitizing and storing of analog speech and video signals to which new hardware and software enhancements can be made without a large investment in resources are stated.

A complete discussion of how the AFIT Speech Lab's Cromemco Z-2, Z-80, S-100 based microcomputer system is utilized to meet these requirements by serving as an I/O channel for the Lab's Nova 2 minicomputer is presented.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

20. ABSTRACT (Continued from reverse side)

The discussion includes detailed descriptions of the hardware developed to connect an 8-bit system to a 16-bit system, the development of the I/O channel communication protocol that allows the two computers to communicate, and the operating system software that provides the control function for the two computers.

END